Spoken Language Translator: Phase Two Report (Draft)

Ralph Becket[1], Pierrette Bouillon[4], Harry Bratt[2], Ivan Bretan[3],
David Carter[1], Vassilios Digalakis[2], Robert Eklund[3], Horacio Franco[2],
Jaan Kaja[3], Martin Keegan[1], Ian Lewin[1], Bertil Lyberg[3],
David Milward[1], Leonardo Neumeyer[2], Patti Price[2], Manny Rayner[1],
Per Sautermeister[3], Fuliang Weng[2] and Mats Wirén[3]

February 1997

SRI Project 6393

This is a joint report by SRI International and Telia Research AB;
published simultaneously by Telia and SRI Cambridge.

[1] SRI International, Cambridge
[2] SRI International, Menlo Park
[3] Telia Research
[4] ISSCO, Geneva

Executive summary

Spoken Language Translator (SLT) is a project whose long-term goal is the construction of practically useful systems capable of translating human speech from one language into another. The current SLT prototype, described in detail in this report, is capable of speech-to-speech translation between English and Swedish in either direction within the domain of airline flight inquiries, using a vocabulary of about 1500 words. Translation from English and Swedish into French is also possible, with slightly poorer performance.

A good English-language speech recognizer existed before the start of the project, and has since been improved in several ways. During the project, we have constructed a Swedish-language recognizer, arguably the best system of its kind so far built. This has involved among other things collection of a large amount of Swedish training data. The recognizer is essentially domain-independent, but has been tuned to give high performance in the air travel inquiry domain.

The main version of the Swedish recognizer is trained on the Stockholm dialect of Swedish, and achieves near-real-time performance with a word error rate of about 7%. Techniques developed partly under this project make it possible to port the recognizer to other Swedish dialects using only modest quantities of training data.

On the language-processing side, we had at the start of the project a substantial domain-independent language-processing system for English, a preliminary Swedish version, and a sketchy set of rules to permit English to Swedish translation. We now have good versions of the language-processing system for English, Swedish and French, and fair to good support for translation in five of the six possible language-pairs. Translation is carried out using a novel robust architecture developed under the project. In essence, this translates as much of the input utterance as possible using a sophisticated grammar-based method, and then employs a much simpler set of word-to-word translation rules to fill in the gaps.

The language-processing modules are all generic in nature, are based on large, linguistically motivated grammars, and can fairly easily be tuned to give good performance in new domains. Much of the work involved in the domain adaptation process can be carried out by non-experts using tools developed under the project.

Formal comparisons are problematic, in view of the different domains and languages used and the lack of accepted evaluation criteria. None the less, the evidence at our disposal suggests that the current SLT prototype is no worse than the German Verbmobil demonstrator, in spite of a difference in project budget of more than an order of magnitude.

Acknowledgements

The Spoken Language Translation project was funded by Telia Nät Tjänster (Swedish Telecom Networks Division) and the work was carried out jointly by Telia Research and SRI International.

We would like to thank Christer Samuelsson for making available to us the LR compiler described in Chapter 6, and Steve Pulman and Christer Samuelsson for helpful comments on the material in 6.

The work described in Appendix A was partly funded by the Defence Research Agency, Malvern, UK, under Strategic Research Project AS04BP44. We are grateful to Małgorzata Styś for comments on the material in that appendix and also for providing her analysis of the Polish examples in it. The greater part of the work described in Chapter 11 was carried out under funding from SRI International (at SRI Cambridge) and Suissetra (at ISSCO, Geneva).

Much of the material in this report is based on published papers. Permission to re-use parts of those papers is gratefully acknowledged. Chapter 2 uses material from [IVTTA96], ©IEEE; Chapter 5 uses Rayner and Carter (1997), ©IEEE; Chapters 5 and 6 draw on Rayner and Carter (1996), and Appendix A uses Carter (1995), which are both ©Association for Computational Linguistics; Chapter 11 is an edited version of Rayner, Carter and Bouillon (1996); Chapter 12 uses Rayner and Bouillon (1995); Chapter 14 is based on Rayner *et al* (1996); part of Chapter 16 uses Carter *et al* (1996) and Chapter 17 uses Rayner *et al* (1994), both of which are ©IEEE.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Robert Eklund, Ian Lewin, Manny Rayner, and Per Sautermeister

1.1 Overview of the project

The Spoken Language Translator (SLT) project has now been running under sponsorship from Telia Research since the middle of 1992. Its long term goal is to produce a realistic system capable of translating human speech from one language into another. A previous report (Agnäs *et al*, 1994) described the results of the first, one-year, phase of the project. The present report will focus on work performed since then, during the period ending in January 1997; however, in order to make the document more self-contained, we will include some material from the earlier report. **Note that the version you are reading here is only a draft, and several chapters still require extensive revision. We expect the final version of the report to be completed not later than March 30, 1997.**

We will start this introductory section by looking at the most basic questions: why we want to build spoken language translation systems at all, what the basic problems are, what we can realistically attempt today, and what we have in fact achieved. Later in the chapter, we present an overview of the main system architecture (Section 1.2) and an example session (Section 1.3). The remainder of the report describes the technical aspects of the system in extensive detail.

1.1.1 Why do spoken language translation?

The SLT project represents a substantial investment in time and money, and it is only natural to ask what the point is. Why is it worth trying to build spoken language translation systems? We think there are several reasonable answers, depending on one's perspective.

In the long term, it is obvious that a readily available, robust, general-purpose machine for automatic translation of speech would be unbelievably useful. It is in fact

almost meaningless to talk about the commercial value of such a device; it would probably transform human society as much as, for example, the telephone or the personal computer. However, it is only realistic to admit that we are still at least 10 or 20 years from being able to build a system of this kind. To be able to maintain credibility, we also want to point to closer and more tangible goals.

In the medium term, it is uncontroversial to state that the field of speech and language technology is growing at an explosive rate. Spoken language translation is an excellent test-bed for investigating the problems which arise when trying to integrate different sub-fields within this general area. It involved attacking most of the key problems, in particular speech recognition, speech synthesis, language analysis/understanding, language generation and language translation. By its very nature, it also requires a multi-lingual approach. These are exactly the reasons which have prompted the German government to organize its whole speech and language research programme around the Verbmobil spoken language translation project.

In the short term, spoken language translation is one of the most accessible speech and language processing tasks imaginable. Technical explanations are not necessary: one just has to pick up the microphone, say something in one language, and hear the translated output a few seconds later. We know no simpler way to convey to outsiders the excitement of working in this rapidly evolving field, and quickly demonstrate the increasing maturity and relevance of the underlying technology. We have been staggered by the media interest which the SLT project has attracted (a summary appears in Appendix B). If people are this curious about what we are up to, we feel that, at the very least, our research must be asking the right questions. We hope that the rest of the report will help convince the reader that we are making good progress towards identifying acceptable answers.

1.1.2 What are the basic problems?

We will now take a step backwards and spend a few paragraphs evaluating where we are with respect to our long term goals in spoken language translation. Many of these are clearly still distant, and will not be achieved within a two- or three-year project. It is none the less important to satisfy ourselves and our critics that we are moving in the right direction.

To fully realize our long term goals, then, we would need to solve nearly every major problem in speech and language processing. On the speech side, we would need to be able to recognize continuous, unconstrained, spontaneous speech in a large number of languages, using an unlimited vocabulary and achieving a high level of accuracy. It would be desirable to be able to do this either over the telephone, or using a hand-held device, or both. We would want recognition to be speaker-independent (no previous training on a given speaker should be necessary), and robust to many kinds of variation, in particular variation in dialect; if we are in a translation situation, we expect at least two different linguistic groups to be present. We would also need to be able to synthesize high-quality output speech. Recognition and synthesis would need be able to take account of both the words that are spoken, and also of the *way* that they are spoken (their prosody) since this often conveys an important component of the meaning.

On the language side, we would need to be able to produce accurate, high-quality translations of unconstrained, spontaneous speech, again including both the actual words and their prosody. In practice, this would probably involve being able to analyse arbitrary utterances in the source language into some kind of abstract representation of their meaning; transforming the source-language meaning representation into a corresponding structure for the target language; and generating target-language translations annotated with the extra information (emphasis, punctuation, etc) needed to allow synthesis of natural-sounding speech. It is highly desirable that the techniques used to perform these various processing stages should be domain-independent, or at least easily portable between domains.

Spontaneous speech is frequently ill-formed in a variety of ways; in particular, speakers can and often do change their minds in mid-sentence about what it is that they are going to say, and speech recognition is certain to fail at least some of the time. Translation must thus be robust enough to deal with more or less seriously ill-formed input. One would also prefer translation to be "simultaneous", in the sense that it should lag a short distance behind production of the source-language utterance. This implies that processing for both speech and language should work incrementally in real-time.

The requirements outlined above are naturally well beyond the state of the art in automatic spoken language translation, and would indeed tax the capabilities of even the most skilled human interpreters. (Anyone who has listened to the simultaneous translation channel at a bilingual conference will testify to this). Some compromise with the current limitations of speech and language technology is necessary. This leads on to our next question.

1.1.3 What is it realistic to attempt today?

If we are to cut the problem down to a size where we can expect to show plausible results using today's speech and language technology, we must above all limit the variability of the language by confining ourselves to a specific, fairly concrete domain. Partly because of the availability of recorded data, we have chosen airline flight inquiries. Other groups working on similar projects have chosen conference registration and meeting scheduling. Domains like these have core vocabularies of about 1000 to 2500 words, which is about all that the current generation of continuous-speech speech recognizers can manage. In view of the non-trivial effort required to port speech and language technology to a new language, it is also sensible to start with a small number of languages.

The speech technology we are using is speaker-independent, and can be run on high-end workstations, either directly or via a telephone connection. Recognition performance is most simply measured in terms of the *word error rate* (WER), roughly the proportion of input words incorrectly identified by the recognizer. Current technology places a lower limit of about 3–10% on the WER for tasks of the type considered here, depending on various factors; in particular, the nature of the communication channel (close-talking microphone *versus* telephone), the degree to which the system is optimized for speed as opposed to accuracy, the language in question, and the amount of training data available.

We expect that continuing advances in hardware will in the near future make it

possible to package sufficiently powerful processors in wearable or hand-held devices, though we have not made any attempt as yet to realise this idea concretely. It also appears quite feasible to aim for systems that permit a high level of dialectal variation, and this, in contrast, is a goal we have actively pursued.

Speech synthesis technology has made great progress over the last few years, and with today's technology it is possible to produce synthesis of fairly high quality. The challenge is now to incorporate natural-sounding prosody into the synthesized output; this is a hot research topic. When determining the correct prosody for the output, it is feasible, though challenging, to try to take account of the prosody of the input signal.

With regard to language processing, it appears that high-quality translation requires some kind of fairly sophisticated grammatical analysis: it is difficult to translate a sentence well without precisely identifying the key phrases and their grammatical functions. All grammars constructed to date "leak", in the sense of only being able to assign reasonable analyses to some fraction of the space of possible input utterances. The grammar's performance on a given domain is most simply defined as its *coverage*, the proportion of sentences which receive an adequate grammatical analysis. It is feasible in restricted domains of the kind under discussion to construct grammars with a coverage of up to 85–90%. Going much higher is probably beyond state-of-the art. Similar coverage figures apply to the tasks of converting (*transferring*) a source-language representation into a target-language representation, and generating a target-language utterance from a target-language representation.

Achieving this kind of performance using domain-independent techniques is once again feasible but challenging. There are a number of known ways to attempt to make language processing robust to various kinds of ill-formedness, and it is both feasible and necessary to make efforts in this direction. Simultaneous translation, in contrast, still appears to be somewhat beyond state-of-the art.

The preceding paragraphs sketch the limits within which we currently have to work. In the the next section, we give an overview of what we have achieved to date during the SLT project.

1.1.4 What have we achieved?

The current SLT prototype is capable of good speech-to-speech translation between English and Swedish in either direction within the airline flight inquiry (ATIS) domain. Translation from English and Swedish into French is also possible, with nearly the same performance. There is an initial version of the system which translates from French into English.

A good English-language speech recognizer existed before the start of the project, and has since been improved in several ways. During the project, we have constructed a good Swedish-language recognizer. This has involved among other things collection of a large amount of Swedish training data. The recognizer is essentially domain-independent, but has been tuned to give high performance in the air travel inquiry domain. There is also a credible first version of a French-language recognizer.

The main version of the Swedish recognizer is trained on the Stockholm dialect of Swedish, and achieves near-real-time performance with a word error rate of about 7%.

Techniques developed partly under this project make it possible to port the recognizer to other Swedish dialects using only modest quantities of training data.

On the language-processing side, we had at the start of the project a substantial domain-independent language-processing system for English, a preliminary Swedish version, and a sketchy set of rules to permit English to Swedish translation. We now have good versions of the language-processing system for English, Swedish and French. There is good coverage for each language within the chosen domain, and fair to good support for translation in five of the six possible language-pairs. Translation is carried out using a novel robust architecture developed under the project. In essence, this translates as much of the input utterance as possible using a sophisticated grammar-based method, and then employs a much simpler set of word-to-word translation rules to fill in the gaps.

The language-processing modules are all generic in nature, are based on large, linguistically motivated grammars, and can fairly easily be tuned to give good performance in new domains. Much of the work involved in the domain adaptation process can be carried out by non-experts using tools developed under the project.

Formal comparisons are problematic, in view of the different domains and languages used and the lack of accepted evaluation criteria. None the less, the evidence at our disposal suggests that the current SLT prototype is no worse than the German Verbmobil demonstrator, in spite of a difference in project budget of more than an order of magnitude[1]. We think we are making good progress in a challenging and topical research area.

1.2 Overall system architecture

The SLT system architecture can best be understood through its fundamental design philosophy which is to combine processing efficiency in any one configuration of the system with relatively easy reconfigurability of the system to new language pairs and application areas. The SLT translation system for given language-pairs in given application areas is therefore configured from *general-purpose* speech and language processing components. Throughout the system there is a basic distinction between *processing engines* and *customization data*. The engines are as far as possible general-purpose "shells"; to be useful for a specific purpose, they need to be supplied with customization data.

The two main engines in the SLT system are the speech-recognition component, the DECIPHER(TM) recognizer, and the Core Language Engine (CLE), designed for the semantic processing of text. We shall first describe each engine in terms of our general architectural principle and then describe their functioning in the overall information flow within the SLT system.

Over-simplifying the picture a little, one can say that the recognizer is a general tool for recognition of speaker-independent, connected speech. It is not tied to any particular language or domain. To port the recognizer to a new language, three basic types of customization data need to be supplied. Firstly, the recognizer requires samples of

[1] At least one knowledgeable and impartial observer who has recently seen both systems claimed that SLT was the better of the two.

the language's basic sounds (roughly speaking, its vowels and consonants). These are recorded by native speakers of the language in question, using enough different speakers to capture common variations in pronunciation. The second piece of data needed is a pronunciation dictionary; this lists several tens or hundreds of thousands of words, together with their valid pronunciations. Pronunciation dictionaries are now available for most major languages. The third main piece of customization data is a few million words of sample text in the language; this most commonly consists of material taken from newspapers, which are often available in machine-readable form. The text material is used to build up a basic *language model*, allowing the system to get some idea of which words tend to follow which; this means that the recognizer can use the preceding words in the current sentence to guess the next one, which in practice greatly increases its accuracy.

To port the recognizer to a specific domain, one also needs a sample of a few tens of thousands of words of dialogue taken specifically from that domain; this material is used to "tune" the language model more closely to the idiosyncrasies of the domain. So for example in the Air Travel Inquiry domain currently being used in the SLT demonstrator, even a small sample is enough to be able to discover that the word "show" is frequently followed by the word "flights", that names of airlines and airports are much more common than in general speech, and so on.

We stress that the above *is* an over-simplification; once the customization data has been provided, a certain amount of manual adjustment by skilled software engineers is still necessary if the recognizer is to achieve a useful level of performance. The effort needed to perform these adjustments is however measured in person-weeks or -months, and is orders of magnitude lower than that which would be required to build a new system from scratch.

The DECIPHER(TM) recognizer is one of the two main processing engines in SLT; the other is the SRI Core Language Engine (CLE), a shell designed for semantic processing of text. The basic idea behind the CLE is to process language by converting it into a uniform logic-based format in which the words have been linked to show the grammatical functions that relate them. This is worth doing for reasons that have been explored by many generations of theoretical linguists. Although languages often appear very different on the surface, at a deeper level they tend to make use of a fairly limited repertoire of grammatical ideas, like "subject", "object", "tense" and so on. By reducing language to its abstract representation, the problems involved in manipulating it are greatly simplified. Translation, in particular, becomes a relatively tractable task, especially when the languages belong to the same family. The particular abstract linguistic representation used by the CLE is known as Quasi-Logical Form (QLF; Alshawi (ed), 1992, Alshawi and Crouch, 1992; see also page 69).

The customization data needed for the CLE to be able to process the sentences from a given language that are likely to occur in the domain(s) to be processed is called a *linguistic description*; this is essentially a detailed grammar and lexicon for that language, written in a format which is based on current linguistic theory and directly usable by the CLE software. A linguistic description for a new language needs to be written by a trained linguist, and doing so is a non-trivial task. We have discovered, however, that the task becomes much easier if a description is already available for a closely related language. Our first linguistic description, written for English, required several person-

years of effort; using this as a base, it was possible to build decent Swedish and French descriptions using about one person-year for each. We are currently implementing a Spanish linguistic description. Because of the close relationship between Spanish and French, we expect this to take less than six person-months.

When the CLE has been equipped with a linguistic description for a particular language, it can be used to convert sentences from that language into their representations in Quasi Logical Form; conversely, the system can take QLF representations and turn them into normal language. Just as with speech recognition, a non-trivial sample of domain text is also required if the CLE is to achieve high performance within a particular application. Since language is generally ambiguous (most sentences have more than one possible grammatical analysis), the system needs a set of examples to show it which analyses tend to be plausible in a specific context. For example, once the CLE has seen a few dozen examples of flight enquiry sentences, it knows that the prepositional phrase *after five P M* in the sentence

Show me flights after five P M

is almost certainly a part of the noun phrase *flights after five P M*, making the sentence mean "show me those flights that are after five P M"; it is most unlikely to be a verb phrase modifier, which would make it mean "Show me some flights, and do it after five P M".

We now describe the information flow in the complete SLT system. The basic flow of processing is as as shown in Figure 1.2. Speech enters the system at the top left of the diagram. For each input, the recognizer outputs a list of the top five sentence strings. The strings are aligned and conflated, thereby generating a speech hypothesis lattice which forms the principal input to the CLE source language processor. A robust bottom-up parsing method is then used to generate a QLF meaning representation.

The diagram indicates a processing stage "plausible QLF extraction" which is one instance of a general processing principle we actually employ at several stages of analysis. At these stages, we extract, using a stack-decoder algorithm (Paul, 1992), the current best sequence of analysis fragments. The criteria for "best sequence" include both the acoustic plausibility scores delivered by the recognizer and the linguistic information acquired during the analysis process. The best sequence is sent over to a target language copy of the CLE, which uses them as input to transfer and generation. Currently, this extraction is performed at four levels, corresponding to completion of the following processing stages: (i) receipt of raw recognizer output, (ii) lexical lookup and tagging, (iii) parsing of small phrases and (iv) full parsing. That is, as soon as one of these stages is complete, we send our best current candidate analysis for transfer. In this way, the translation process can produce a first rough version of a translation very quickly, and can then refine and improve that translation as further processing of the source language takes place. The final level of extraction – extracting the most plausible source language QLF from a full parse of the input – is the stage indicated in the diagram. The result of translating this represents our best effort translation, utilizing all the knowledge sources available to the system.

The transfer process itself is performed using two different methods. The first method, which is applied at stages (i) and (ii), uses rules that directly associate word sequences (optionally tagged by part-of-speech) in source and target languages. The

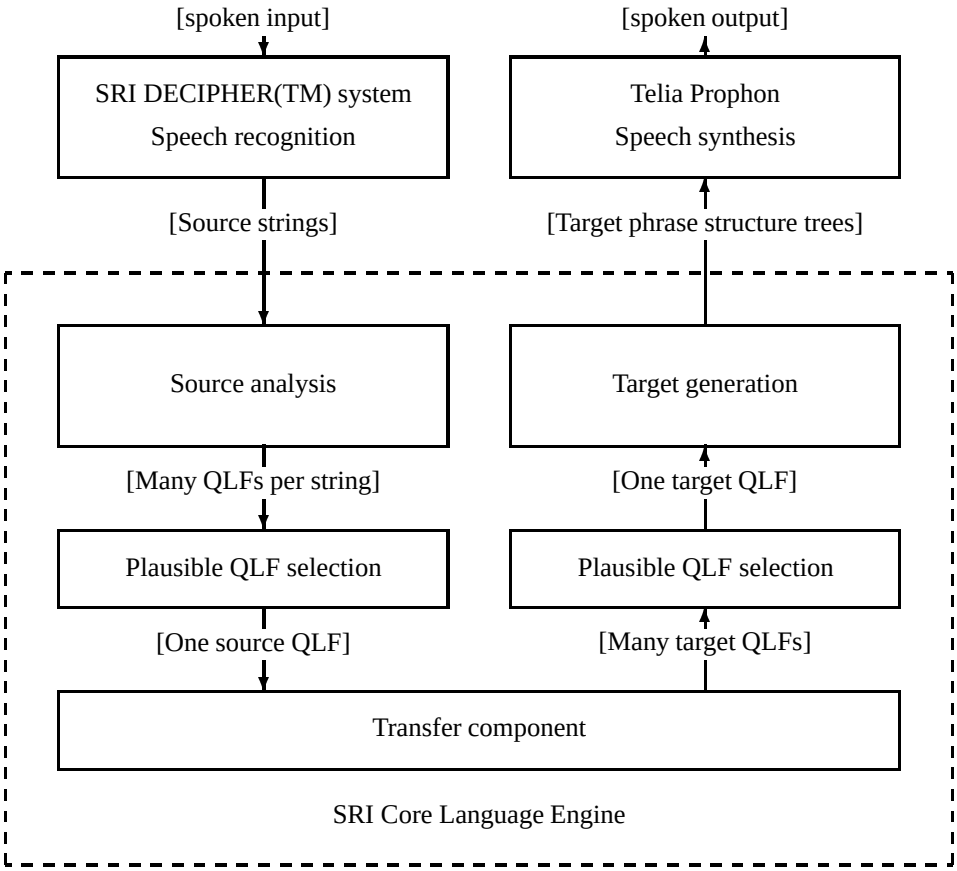


Figure 1.1: Basic SLT processing

second, applied at stages (iii) and (iv), uses unification-based rules, operating on source language QLFs, to suggest candidate target language QLFs from which target language strings are generated using the generator described in Alshawi (1992, pp268ff). In both methods, statistical preferences are used to choose between competing candidates. The basic division of effort is that rules encode domain-independent grammatical and lexical phenomena, while the preferences take care of problems concerning lexical choice, which are usually to some extent domain-dependent.

If the unification-based method is able to produce a translation at all, it is generally of high quality. However, it is only applicable to fragments produced during the last two stages of processing, when at least some grammatical information has been applied. Also, like most rule-based procedures, it tends to be somewhat fragile. This is why we use the less sophisticated surface translation (word-sequence based) method at the first two stages. The bilingual phrasal lexicon which encodes the required information is built semi-automatically using a simple corpus-based tool.

The two translation methods both add their results into a “translation chart” which initially mirrors the source language chart used by source language processing. At any point in translation, it is possible to pause and extract a current best sequence of translation fragments from the chart. This extraction is performed using the same stack-decoder algorithm as is used on the analysis side. It is therefore simple to impose time-limited translation – as soon as any source language processing at all has been received, we can generate a candidate translation, and at any point after that, we can generate a current best estimated translation.

The visible result is that the translation process produces a first rough version of the translation very quickly, using the surface method; it then refines it over several iterations as edges produced by the deep translation method become available. When no more edges are available for processing, or alternately when a pre-set time-limit has been exceeded, the best sequence of translation fragments for the final version of the chart is extracted and sent to a speech synthesizer. Speech synthesis is handled by the Telia Prophon system for Swedish, and by the CNET TTS system for French.

1.3 An SLT session

In this section, a typical SLT demonstration will be described. It will try to serve as a small “demo on paper”, to give an impression of the system’s capabilities.

An SLT demonstration typically begins with a short talk, given by the demonstrator, where the basics of speech recognition, automatic text translation and speech synthesis are explained at a level deemed appropriate. After the talk, the demonstrator utters a sentence to be recognized, translated and synthesized. What the spectators see is shown in Figure 1.2, where the Swedish sentence “lista de billigaste flygningarna mellan Boston och Atlanta med mellanlandning i Philadelphia på fredag eftermiddag” is translated into its English counterpart “list the cheapest flights between Boston and Atlanta with stopover in Philadelphia on Friday afternoon”. The interface is either viewed on the computer screen or projected onto a larger, wall-hanging, screen.

On the present demonstration computer at Telia Research, Haninge – a Sun Ultra 2 – recognition is virtually instantaneous, translation takes approximately 15 seconds for the sentence in Figure 1.2, and American English synthesis (TrueTalk) takes around two seconds.

Speech translation in the SLT project is a dynamic process, and the interface brings forth much of this process. The interface shown in Figure 1.2 will now be described in detail.

1.3.1 The Interface Elucidated

In Figure 1.3 the Swedish sentence “Jag vill ha lunch” has been translated into its equivalent English utterance “I want lunch” (total time required around five seconds). As was said above, the translation is a dynamic process and things “happen” on the interface. The different controls, buttons and labels will here be explained one by one in the order they appear on the interface.

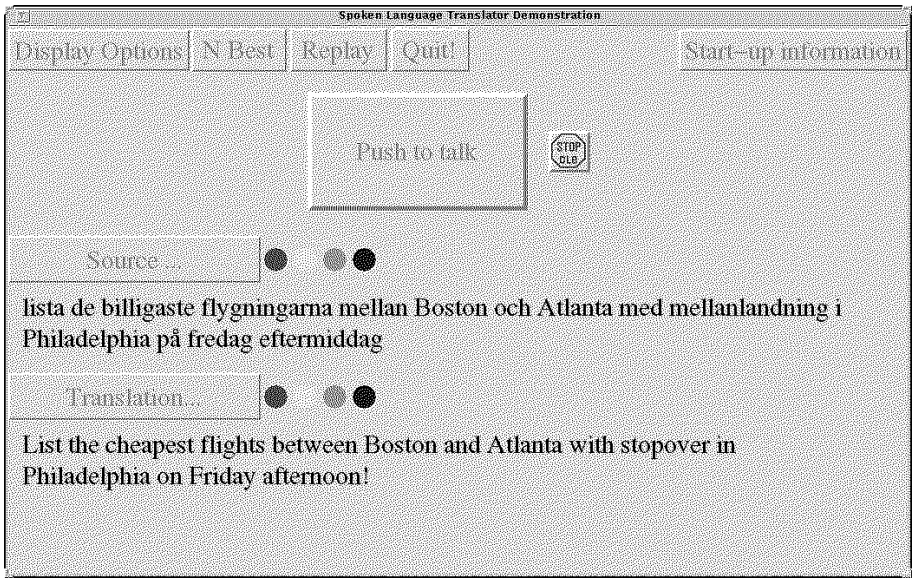


Figure 1.2: Example of demo session window. The Swedish sentence “lista de billigaste flygningarna mellan Boston och Atlanta med mellanlandning i Philadelphia på fredag eftermiddag” is translated into English “list the cheapest flights between Boston and Atlanta with stopover in Philadelphia on Friday afternoon”. The time required on an Ultra 2 is around 25 seconds, from the beginning of the Swedish utterance until the end of the English utterance.

Display Options This button provides the demonstrator with a menu of options. One can either choose between `Executive` or `Detail` modes. The former is shown in Figure 1.2 and Figure 1.3. The `Detail` mode shows more of the process explicitly, as is shown in Figure 1.4, where the Swedish sentence “jag skulle vilja boka en limousine till city” is translated into its English translation “I’d like to book a limousine to downtown”. Here the differences between the most preferred strings at the levels of the recognized surface form (S), the lexical lookup level (L), the phrase level (P) and the full sentence levels (F) (these are the levels (i) to (iv) we have already met on page 7) are clearly shown. A further choice is to show CLE internal boundaries in the sentence shown underneath the `Translation` label. One can also choose between three different font sizes, small, default and large. Default size is shown in the figures in this section.

N Best This button is pushed to display a the N-best list produced by the recognizer. The number N is presently five (see Section 16.1 for the reasons for this). A pedagogical point here is that the demonstrator can show that the first alternative may not be the correct one, and that, for example, higher levels of processing have discarded one or more hypotheses in favour of one with lesser acoustic probability for grammatical reasons. Thus, the outputted sentence might be number three

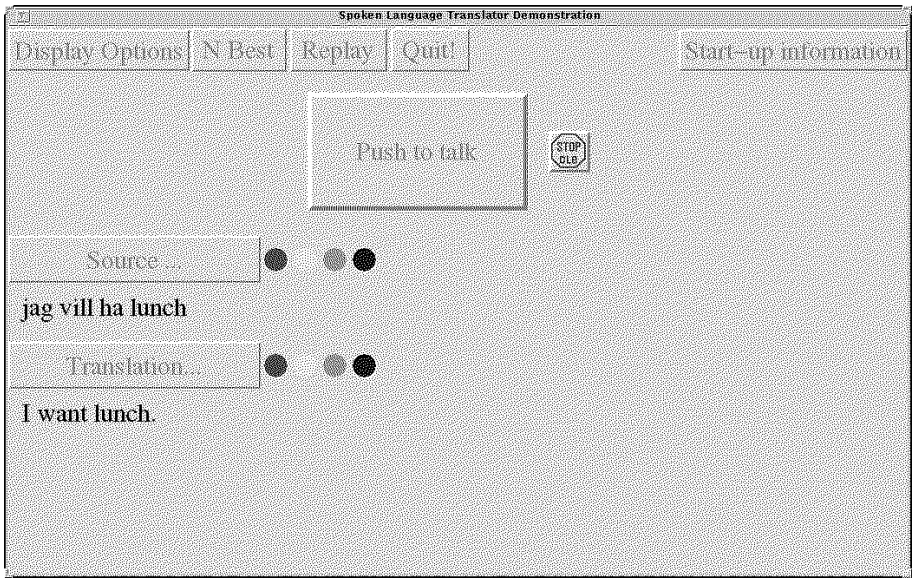


Figure 1.3: Example of demo session window. The Swedish sentence “jag vill ha lunch” is translated into its English equivalent “I want lunch”. Time required on an Ultra 2 is approximately five seconds from the beginning of the Swedish utterance until the end of the English utterance.

in the N-best list.

Replay This button is pushed in order to play the synthesized translation again.

Quit This button ends the session.

Start-up information This button tells the demonstrator when the different modules are all set for demoing, i.e., it shows when the recognizer is ready, as well as the two language modules (source and target language). It is thus used prior to a demo session, rather than as a part of the demo.

Push to talk This button is pushed when the demonstrator speaks to the system, i.e., it activates the system.

STOP CLE This button is pushed in order to interrupt linguistic processing if need be.

Source ... Underneath this button-like sign the recognized source language string appears upon recognition. On the present demonstration computer, a Sun Ultra 2, recognition occurs in close to real time. The four “lamps” on the right-hand side of the label indicate different stages in the translation process, and correspond to the following parts of the process, respectively:

The first (leftmost) lamp (red) This lamp indicates pure word recognition. It corresponds to the letter “S” in the `Detail` mode.

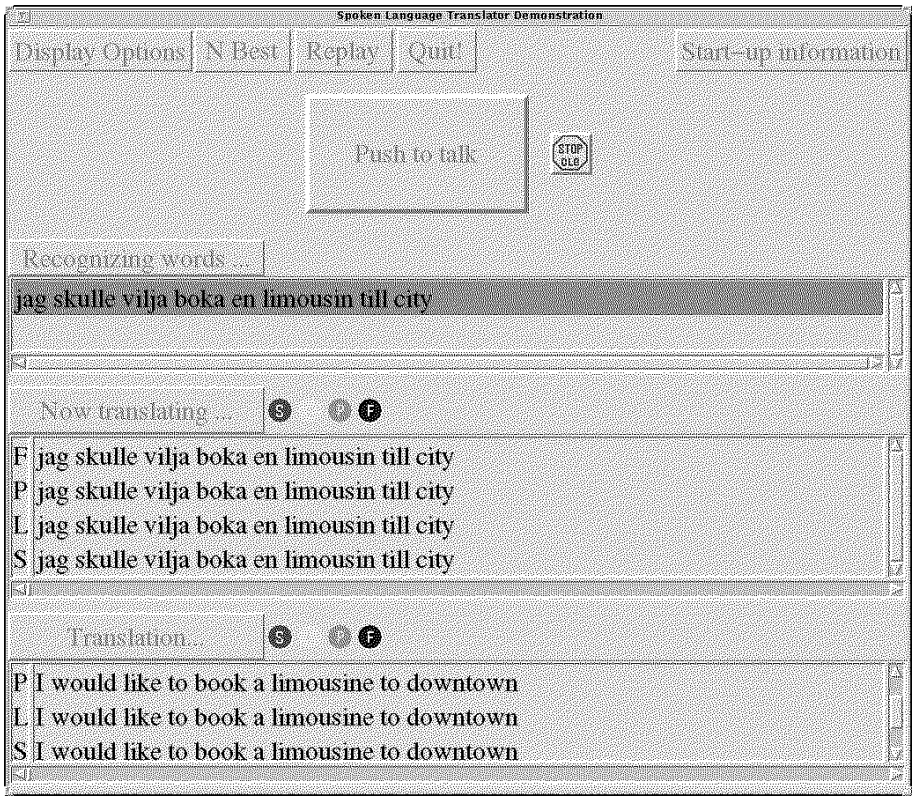


Figure 1.4: Example of demo session window with `Detail` mode set. The Swedish sentence "jag skulle vilja boka en limousine till city" is translated into its English equivalent "I'd like to book a limousine to downtown". Time required on an Ultra 2 is approximately thirteen seconds from the beginning of the Swedish utterance until the end of the English utterance.

The second lamp (yellow) This lamp is lit when lexical lookup is applied. This means that simple, grammatical information is attached to the hypothesized word. It corresponds to the letter "L" in the `Detail` mode.

The third lamp (green) This lamp indicates when phrase level knowledge is used, i.e., larger grammatical chunks such as "in Philadelphia", "want to fly" and so forth. It corresponds to the letter "P" in the `Detail` mode.

The fourth lamp (blue) This lamps shows when entire sentences are looked at and processed. It corresponds to the letter "F" in the `Detail mode`.

Translation ... Underneath this label the translated sentences appear. This typically takes a couple of seconds for a short sentence like "Jag vill ha lunch" ("I want lunch"), and 15 to 20 seconds for longer sentences like "lista de billigaste flygningarna mellan Boston och Atlanta med mellanlandning i Philadelphia på

fredag eftermiddag som serverar lunch" ("list the least expensive flights between Boston and Atlanta with stopover in Philadelphia on Friday afternoon that serve lunch"). As is the case with the recognized sentence, there are four lamps to the right of the sign that indicate different stages in the translation process. Since robust translation is used (see Sections 1.2 and 5.2), a translation is always available, but it typically changes as higher levels of knowledge are applied.

1.3.2 Synthesis

The synthesizers used are all state-of-the-art, and provide high-quality speech of great naturalness. Currently, they are:

Swedish: Prophon (Bäckström *et al*, 1989), a concatenation synthesizer based on polyphones, in this case demi-syllables. It uses a female voice.

English: a licenced version of TrueTalk from Entropics (Entropic Research Laboratory, 1995). It is also a concatenation synthesizer, based on diphones.

French: a licenced version of CNETVOX. Once again, it is a concatenation synthesizer, based on polyphones (ELAN Informatique, 1996).

1.3.3 Final Comments

This section has tried to introduce the SLT system in a synoptic, "crash-course"-like way. Of course, there is much more behind the scenes, and the different components will be described in detail in the following chapters.

As compared to the previous SLT-1 demonstration tools, much has been done in order to provide the spectator with a more nice-looking and more pedagogical interface. Emphasis has been put on a clear and attractive-looking display to demonstrate and explain both the overall SLT process and its different sub-components.

To end this introductory chapter on an international vein, an example of Swedish-to-French example is given in Figure 1.5. The Swedish sentence "lista de billigaste flygen till Philadelphia" is translated into French "indiquez les vols les moins chers à destination de Philadelphie".

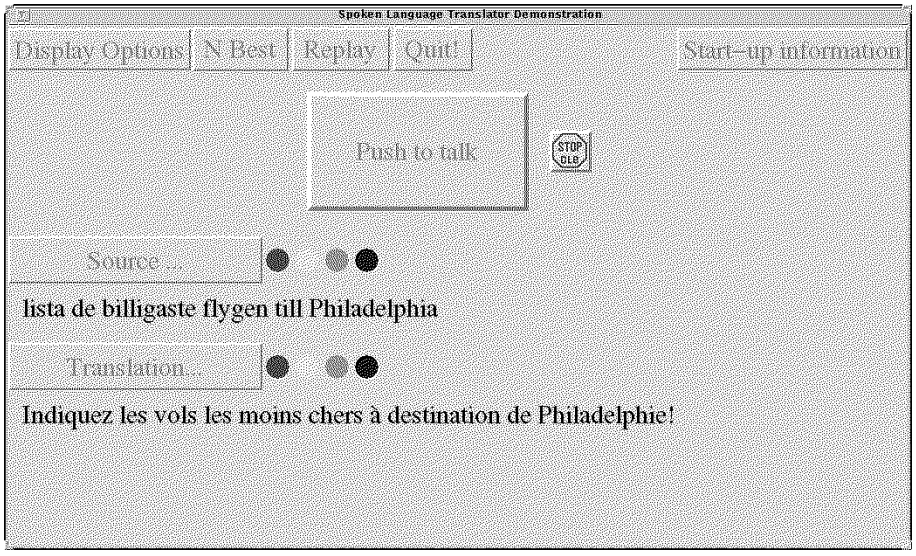


Figure 1.5: Example of demo session window. The Swedish sentence “lista de billigaste flygen till Philadelphia” is translated into French “Indiquez les vols les moins chers à destination de Philadelphie”. Time required is around 7 seconds from the beginning of the Swedish utterance until the end of the French utterance.

Chapter 2

Language Data Collection

Robert Eklund, Ivan Bretan and Catriona MacDermid

In this chapter language data collection is discussed. Different methods to collect language data are described and compared. Both quantitative and qualitative aspects are considered.

2.1 Rationale and Requirements

DECIPHER (Murveit *et al*, 1993) has been trained for the ATIS domain using data collected in a large-scale Wizard-of-Oz (WOZ) simulation by the MADCOW (Multi-site Atis Data Collection Working) group (Hemphill *et al*, 1990) to be described below. These data are needed both to train the acoustic-phonetic model of the speech recognizer, the lexical and *n*-gram language model. In addition, they can be used to streamline the development of the linguistic modules of the system, in particular the lexicon, grammar, set of collocations, transfer rules, and dialogue model (if existing). This streamlining can be achieved rationally by means of constructing representative corpora, where utterances are sorted according to the frequency of their syntactic pattern (Rayner *et al*, 1995; see also Chapter 8). In a corpus-oriented development framework, the quality of the system is dependent on the quality of the corpora used. Thus, in SLT, significant efforts have been devoted to obtaining high-quality linguistic data. A question which must be answered before embarking on such an undertaking is what the measures of quality are. One could imagine that genuine human–human conversations would provide the best yardstick for linguistic training data, but this is not necessarily true given that the linguistic performance models for users engaging in dialogue with a machine varies with the behaviour of the system (Bretan *et al*, 1995). Thus, the data obtained from human–human dialogues do not transfer straightforwardly into the design of human–machine dialogues. There is also the ethical issue of "bugging" people's conversations, which is most critical if the speakers can be identified and if the dialogue contains sensitive or personal information. Ideally, informed consent should then be obtained.

Turning instead to data collection methods geared toward human–machine dialogues, there are two principal alternatives: WOZ simulations and "bootstrapping". WOZ simulations (see Section 2.2.1 below) are carried out by setting up a scenario where the user interacts with an alleged speech-understanding system partly or fully simulated by a human, exhibiting the type of capabilities that the system to be constructed ideally should possess.

Bootstrapping, in this context, is the procedure of constructing a rudimentary version of a speech-understanding system with a minimal vocabulary (which may be collected through a very limited WOZ study or other less costly sources and complemented by means of manual inspection), and putting it into use "prematurely". This system will obviously not have the intended coverage, but will still elicit many user utterances and words that were previously not recorded. These additional data can be used to improve the system's recognition and understanding capabilities, whereupon new user sessions can be initiated to collect even more data, and so on. It is not clear how useful the linguistic material thus obtained really is, especially in comparison with a WOZ simulation. This is also highly dependent on the quality of the bootstrapped system. One problematic issue in this context is the fact that users adapt to the quality and language of a system over time, which could mean that subsequent versions of the bootstrapped system are more unsophisticated than necessary. This is probably particularly true for systems where complete dialogues need to be recorded in order to obtain the required data, as opposed to one- or two-shot interactions. Life and Salter (1996) claim that bootstrapping and WOZ are both needed, since they are complementary. The WOZ studies give input indispensable for dialogue and interface design, whereas bootstrapping methods are more useful in generating large amounts of data for training acoustic and lower-level language models.

WOZ simulations do have drawbacks – most notably they are laborious, time-consuming and costly. An informal figure quoted in the work on collecting ATIS data estimates the cost of collecting 10,000 sentences to USD 1 million, i.e., $100 per sentence! Although the WOZ simulations we have conducted ourselves for Swedish indicate much lower costs, it is clear that this way of collecting data is expensive. The question then arises whether it is worth initiating a new WOZ simulation for each new language added to SLT. As a cheap substitute approach we have instead been experimenting with "piggy-backing" on the existing American data through textual translation by native Swedish speakers. The resulting data served as the language model for the Swedish version of SLT. One assumption underlying such a project is that spoken utterances can systematically be translated textually on a large scale, preserving the idiomatic traits of spoken language. This assumption proved wrong, but the effort still turned out to be worthwhile, partly from reasons similar to the ones that Life and Salter quote.

Since the email study still did not provide good enough data, a WOZ study was conducted.

2.2 Methodology

In this paragraph, general methodology of data collection is discussed. A number of different methods used during SLT will be described, and a quantitative and qualitative analysis of the different methods will be carried out. A shorter version of this analysis is found in Bretan, Eklund and MacDermid (1996).

2.2.1 Wizard-of-Oz Simulations

In order to collect more realistic training data for a spoken dialogue system, experimental subjects can be recorded as they conduct task-oriented dialogues with a simulated dialogue system. The subjects are most often led to believe that their dialogue partner is a prototype system, when in fact an accomplice (the "wizard") is simulating an operational system by performing some or all of the system's functions. Typically, the wizard interprets the subjects' utterances, simulating speech recognition and often language understanding. Other functions, such as dialogue management and speech synthesis, can be handled by a computerized tool operated by the accomplice (Amalberti *et al*, 1993). Since the subjects have no real task they wish to complete, they are given scenarios describing a given task. Written scenarios are most common, although these can act as a "script", strongly influencing the subject's vocabulary and syntactical structures, at least in their opening utterance within the dialogue. To overcome these limitations, the scenario can be presented in tabular or graphical form, such that the subject has to interpret the scenario using their own words. In this case, the illustrations must be unambiguous for the subjects. In yet another method, subjects are given scenarios in picture form (MacDermid and Goldstein 1996), to be described in Section 2.5 below.

2.2.2 American ATIS Simulations

One example of a WOZ simulation (but with text instead of spoken output from the simulated system) is MADCOW (Hemphill *et al*, 1990). A large number of subjects (2,724) were given written scenarios and spoke to what they believed to be a working system, when in fact human wizards were interpreting the subjects' questions, querying the database by hand, and displaying the results on the subjects' screen. After several months, once enough data was acquired in this way to train the system, the system became fully operational, and the wizards were no longer needed. The expectation was that, because the users believed that they were speaking to a real system, the wizard data and real data were equivalent.

2.2.3 Swedish ATIS Simulations

In order to obtain high-quality data, a spoken language translation system in the ATIS domain was simulated at Telia Research. This was done without the use of any computerized simulation tool. Subjects believed that the "system" translated their Swedish inquiries to an English-, French- or German-speaking travel agent in order to book flights. In fact, no translation occurred in these dialogues. The subject's utterance was

conveyed – usually verbatim – by a wizard (W1), a professional actor, representing the subject's translation system, to a second wizard (W2) representing the travel agent's translation system. The reason for using two wizards was to reduce cognitive overload on the wizard(s). Certain simplifications were made to complex utterances, that is, utterances that were not understood or that were longer than twenty words were rejected by W1, who asked the subject to repeat or reformulate the utterance. The utterance was then conveyed by W2 over the telephone to a Swedish "accomplice" (the "travel agent"), who asked for additional parameters where necessary to complete the booking. W2 then conveyed the travel agent's replies to the subject via W1 according to the same constraints. The two wizards sat in the same room and when they spoke to each other to convey utterances or clarify internal misunderstandings, W1 suspended the microphone contact with the subject. Similarly, W2 used the secrecy button on the telephone. The wizards listened to the respective dialogue partners through headsets. The actor was trained to speak to the subject with the unnatural prosody characteristic of composite digitized speech and had no script apart from the requests to reformulate or repeat. The "travel agent" used certain standard phrases based on an interview with a real travel agent and had access to a paper database constructed with data from an ATIS-type database used in the travel agency. Otherwise, his speech was spontaneous in response to the subject's queries, which were based on a combination of written and graphical scenarios. The dialogue between the subject and the actor was recorded as sound files on a Unix work station and all four input channels were recorded on a DAT-recorder for later transcription. The set-up is shown in Figure 2.1.

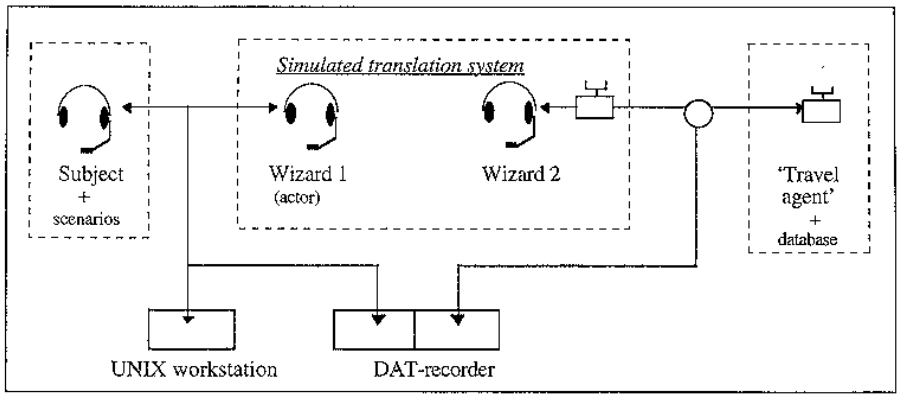


Figure 2.1: Experimental set-up for Swedish ATIS simulation.

2.3 Translations of American WOZ Material

One way to obtain data without having to design WOZ simulations is to translate existing WOZ data collected in another language. During the SLT project, translations of the American WOZ data described above have been used for training and language

modelling purposes. These translations will be described in the following.

2.3.1 Translations... A First Step

During the SLT-1 project, 4,021 American WOZ sentences were translated into Swedish by a bilingual secretary. This corpus will be referred to as C1. Some 10% of the sentences were lost in a disk crash, and were retranslated by a member of the SLT team. This corpus was taken as the basis for the training of the Swedish language model. However, it was quickly realised that the quality of the translations exhibited several flaws.

First, the translations were clearly influenced from the English originals, resulting in "Swedish" most people would find far from idiomatic, or worse.

Second, the lexical coverage was clearly narrow, since the translations fell into a mode, where "standard patterns" recurred to a large extent, beyond doubt caused by the fact that – apart from a few hundred sentences – single person had been involved in the translations. In order to remedy at least the latter of the aforementioned problems, two members of the Swedish SLT team translated a further 3,264 and 1,888 sentences, respectively. THe SLT member who had done the translations of the original corpus also added a further 220 sentences. In order to obtain more idiomatic data, these translators were instructed to use freer translations. In this way, a corpus of 5,372 translated sentences was obtained. This corpus will be referred to as C2.

However, there are several, more general, problems associated with translations.

First, it is very hard for translators to avoid linguistic bias caused by the wordings in the source language, and translations are almost certain to be influenced by expressions idiomatic in the source language but not in the target language.

Second, phenomena like hesitations, repairs, false starts and so forth are not readily translated in a natural way.

Third, translation is likely to miss certain features typical of spontaneous speech, like agreement errors, especially if the target language does not make use of agreement in the same way as the source language. If the target language has different grammatical granularity than the source language, translators will not add grammatical errors that do not exist in the source language utterances.

Fourth, a small number of translators – four in this case – are not likely to be able to provide as great linguistic variability as will the would-be users of the system in question. Thus, it was still felt that the quality of the material was suffering from the translation process, lacking idioms and expressions typical of Swedish.

2.3.2 Email Corpus

Since a WOZ simulation at this point was still out of reach, it was decided to do "more of the same" and spread the translation task between as many translators as possible, with distinct instructions to do *free* translations. To this end, lists of email addresses of Telia employees were obtained. These lists were used for distributing the translation task. The addressees were each sent two emails. The first email contained background information about the task, and an explanation as to what the second email was for.

The first email also provided the addressee with instructions as to how to approach the task. These instructions are briefly outlined below.

Instructions

The addressees were told to avoid "literal" translations. Instead, they were instructed to ponder the "meaning" of the English sentences and then write down what they would have said themselves in Swedish if they were to want the same information in real life.

They were also told to imagine themselves communicating with a computerized system. Therefore, they probably should avoid using too much slang, since they probably would not use too colloquial a language when addressing an automatic, non-human, system.

The risk of translating too literally was once again pointed out by referring to certain expressions and specific wordings. The example given was the English "ground transportation", which could easily be translated literally as "marktransport", which is also the professional term used in Swedish. However, most non-professional Swedes would probably say things along the lines of "förbindelser" ("connections") or "flygbussar" ("shuttle buses").

The addressees were also told to neglect sentences they found totally incomprehensible or "absurd" in one way or another.

The format of the subsequent email, containing the sentences, was elucidated with an example. the addressees were informed that the subsequent email would contain pairs of sentences in the following way:

```
<e5233> Please list all flights between Atlanta and San Francisco
<s5233>
<e3665> Is booking necessary?
<s3665>
```

The addressees were asked to fill in the empty, Swedish, lines with free translations, following the tips given in the letter. An example was given, thus:

```
<e5233> Please list all flights between Atlanta and San Francisco
<s5233> Lista alla flygningar mellan Atlanta och San Francico, tack
<e3665> Is booking necessary?
<s3665> Måste man reservera?
```

Finally, upon finishing the task, the addressees were told to send the email back by using the "Reply" button in their email client.

A contact (address and telephone number) was also given, if the addressees should want to ask something in connection with the task.

First Batch

The procedure described above was executed twice. In the first batch, 11,232 sentences from the American ATIS material were divided into files of 18 sentences each and

emailed to 624 Telia employees. The recall here was 1,116 sentences, i.e., about 10%.

Second batch

Since it was judged that 18 sentences perhaps were a few too many to look attractive, a second batch of 7,533 sentences were divided into files of only 3 sentences each and emailed to 2,511 Telia employees. Recall this time was 1,080 sentences, i.e., about 14.5%.

Even more translations...

An additional 500 sentences were translated by five translators, SLT team members and graduate students of linguistics, the latter of whom working for free, hereby acknowledged.

In this way, a corpus of 4,595 sentences, translated by approximately 427 persons, was compiled.

2.3.3 Sundry Comments on Email Corpus Editing

Before ending this section, some reactions of anecdotal character need to be mentioned lest they go unnoticed. Obviously, a motley collection of reactions was the result of sending out more than 3,000 emails.

First, although all the addressees, by definition, were in possession of email clients, several did not know how to handle them properly. This meant that, instead of getting all the translations in the desired, aforementioned, format, some translations arrived in "hard copy" format, i.e., normal letters, some of whom in all-capitals lettering, with the index numbers omitted, and without a sender. Hence the approximate figure 427.

Since a script, rather than a mailing list was used, the addressees could not see that they were given the task together with hundreds of other Telia employees. Thus, some people of foreign origin called and asked why *they* had been chosen, since they hardly knew Swedish.

Only one person reacted "aggressively", out of approximately 3,140 addressees. The reaction consisted of a couple of annoyed emails, that were responded to in a courteous and explanatory way.

A few persons took the instructions too literally, and translated *too* freely. Thus, "Chicago" was translated into "Malmö", probably caused by the instruction "The way *you* would say it" interpreted by persons who never travel to the USA.

2.4 A Comparison of the Corpora

The different corpora thus collected may vary according to several parameters, such as lexical size, grammatical coverage and idiomaticity, i.e., the use of idiomatic expressions specific to the domain and language. In all the comparisons, C1 and C2 were merged into one corpus, **TC** (four translators: the bilingual secretary of SLT-1 and three SLT team members). TC contains 4,021 + 5,372 = 9,393 sentences. The email corpus

will be referred to as **EC** (approx. 427 translators). A small Wizard-of-Oz pilot of 127 sentences will be called **WOZp** (10 subjects), whereas the WOZ simulation described above will be called **WOZ** (52 subjects).

One issue to be examined here is lexical representation. TC contains 1,581 lexical entries (types). Here, inflected forms etc. are counted as different types. EC contains 1,789 entries, WOZ contains 977 entries and WOZp 174 entries. Fig. N shows how the lexicon grows as a function of the number of collected sentences. As is seen, the lexicon grows most rapidly in EC, whereas the growth rate is more or less the same for TC and WOZ. This seems to indicate that a fast way to obtain good lexical coverage is to involve many persons in the gathering of data for the target language. Another consequence of this is probably reflected in the percentage of words in the lexicon that occur only once in the corpus. In TC and WOZ, around 10% of the lexical entries have only one token, whereas the corresponding figure for EC is 17%.

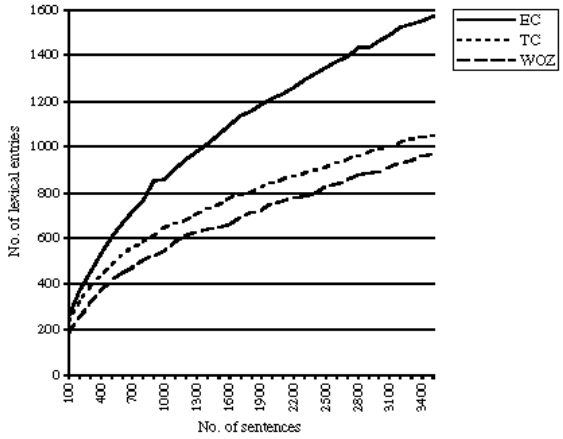


Figure 2.2: Lexicon growth as a function of number of sentences. To avoid sequential effects the data were collected several times in different orders and later averaged.

It does not follow from the fact that EC has a larger number of lexical entries than WOZ that EC is the most representative of Swedish usage, due to the aforementioned problems of “colouring effects” and the lack of speech-specific phenomena associated with textual translation. In fact, the two lexica vary in several respects. First, there are words and constructions that exist in both lexica, but whose frequency is quite different. As an example, the word “okay” exists in both lexica, but is far more common in the WOZ material (sic!). It exists in 3.8% of the sentences in WOZ and 2.5% in WOZp, but only in 0.2% of the sentences in EC. The opposite is equally true. The word “vänlig” (literally “friendly”) is far more common in EC (4.8%) than in WOZ (0.2%), probably as an effect of “please”, a word lacking a good counterpart in Swedish. What Swedes would say is “tack” (“thank you”) in sentence-final position. Thus, “tack” occurs in 20% of the sentences in WOZ but only in 2% of the sentences in EC. The corresponding figure for WOZp is 11.5%. Since these words have different syntagmatic properties,

	WOZp	WOZ	TC	EC
No. of sentences.	127	3,578	9,393	4,595
No. of subjects/translators.	10	52	4	c.427
No. of lexical entries (types)	174	977	1,581	1,789
Percentage of lexical entries with only one token	(–)	c.10	c.10	c.17

Table 2.1: Overview of quantitative data for different corpora.

they also influence grammatical structure. Second, some words and/or constructions common in WOZ do not exist at all in EC. A striking example is that 7% of the sentences in WOZ begin with the word “då” or “ja, då” (fillers that roughly translates as “well”), but not at all in EC. The corresponding figure for WOZp is 0.6%. The idiomatic expression “det går bra” (literally “it goes well”, i.e., “that’s fine”) occurs in both WOZ and WOZp (about 0.7%) but not in EC. Similar examples of skewed material abound, most of which can be accounted for in terms of linguistic bias associated with the translation process. However, it must be borne in mind that the different set-ups for the American ATIS simulations and the Swedish WOZ simulations beyond doubt influenced the linguistic material obtained.

2.5 Concluding Remarks

One problem with the email approach is that disfluent or “strange’ sentences are less likely to be translated than “normal” sentences, since the former require more effort from the translator. This means that the method might act as a filter where marginal sentences become underrepresented in the translation process. The costs of the method are hard to judge, since the work is very much “hidden”. More than 420 persons worked approximately 10–30 minutes each, on a voluntary basis at no cost to the project. The bulk of the work consisted of editing the returned files, many of which did not arrive in the desired format.

Although the email approach produced useful data and translators were instructed to respect source language disfluencies and spoken language style, the results differed from the data obtained in the Swedish WOZ simulation in that certain features of natural speech were notable by their absence in the former corpus. These differences can be attributed partly to a loss of naturalness in the translation process but more importantly to the fact that typical spoken language phenomena (Tannen 1982; Linell 1981; Fromkin 1980) are very specific to language and modality and cannot be obtained through literal translation of text. One way to circumvent this problem is to record oral translations from sources other than text. A method where this is used is the Storyboard method. In this method, subjects are given picture or “storyboard” scenarios and asked to formulate an equivalent utterance (MacDermid and Goldstein 1996). In this way, linguistic bias from written scenarios is avoided. The data is gathered as speech rather than in written form, adding realism, though subjects are not in a “live” dialogue. Consequently, the method is best suited to collecting an initial set of utterances and is

a good way to tap possible variations in use of syntax and vocabulary.

In conclusion, a general recommendation would be to use WOZ simulations to obtain natural speech data, complemented by the email approach – or a similar method – where the task is distributed among a large number of people proficient in both languages to obtain wide lexical coverage. WOZ simulations not only provided proper dialogue data, but also idiomatic spoken language. The textually translated utterances gave no information concerning dialogues (since they were translated one by one), and clearly deviated from normal spoken Swedish. However, the translated corpus contained a much richer variation in vocabulary than the WOZ data: the vocabulary grew quicker. Thus, in addition to the observation that the growth rate of vocabularies varies from domain to domain, made e.g. by Hetherington and Zue (1993), we note that growth rates vary even among vocabularies collected within the same domain. This difference is not surprising, and is mainly attributable to the difference in number of speakers, and to a lesser extent task formulation and modality. There were other differences too, both stylistic and syntactic, but the lexical variation was the most striking phenomenon.

Achieving lexical completeness in data collection is of course decisive when it comes to tuning a speech-understanding system to optimal performance within a given domain. Missing words can lead to system behaviour that slows down, confuses or even misleads a user. Hetherington and Zue also point out that vocabulary completeness is not best achieved by tapping huge generic corpora: 100,000 words from the WSJ corpus are needed to obtain the same coverage of ATIS as a vocabulary of 300 words derived from 2,000 ATIS training sentences! Over-sized vocabularies are of course also sources of performance degradation. Collecting utterances representative to the domain is therefore of utmost importance to the lexical competence of the system.

Chapter 3

Speech Data Collection

Robert Eklund and Jaan Kaja

This section will deal with the collection of speech data. Some of the issues and fields discussed in this chapter are also described in Chapter 4.

3.1 Rationale and Requirements

There are many different parts in the creation of a speech recognizer for a given language, even when the actual recognizer as such already exists, as was the case within the SLT-2 project. The tasks at hand are mainly the following:

First, one needs to collect a large amount of speech data, i.e., recordings of Swedish speakers from different parts of Sweden. To that end, it is necessary to obtain large amounts of texts in computer-readable format.

Second, there is the need to organise and administer recordings on various locations in Sweden. It was immediately felt that the recordings should take place on location, since people have a very marked tendency to adapt to the environment, and had the speakers been brought to Stockholm, there is great likelihood that they would have altered their idioms towards Stockholmian. Therefore, a problem that needs to be solved is to find good enough recording facilities, e.g., rooms and the like, since there is a huge lack of anechoic chambers in Sweden.

Third, one also needs to define what dialect areas need to be covered, and to find recording facilities in the said areas.

Fourth, there is the need to find speakers adhering to the requirements set up within the project (i.e., the speakers should be 'locals', aged between 15 and 65 and so on). Finally, the recorded material must be verified prior to training, lest the recognizer be trained on corrupt material.

Fifth, a lexicon must then be created, fulfilling the requirements of the speech recognizer and covering the texts used in the recording sessions.

Sixth, a further thing to decide on is the kind of hardware equipment one needs to do all this.

All the above will be described in the following paragraphs.

3.2 Text Material

Within the SLT project, the required text material mainly fall into three major categories:

1. Phonologically and phonetically “balanced” material, read by all speakers.
2. Texts within the ATIS domain, i.e., Swedish ATIS sentences.
3. General newspaper/fiction texts, in large enough amount to be able to present unique material to each and everyone of the speakers.

A prerequisite for doing data collection is to have computer-readable texts. One alternative would be to create the text oneself, which may – and probably must – be done concerning the phonologically balanced material, but is clearly out of reach when it comes to the huge amounts of other texts necessary for extensive training like the one carried out within the SLT project. Mainly, two kinds of text were needed. First, Swedish ATIS material was required. Second, general texts – newspaper or general fiction – was needed. The former of these two materials was needed for the domain-specific training, the latter for general training and good coverage of Swedish phonetic material.

3.2.1 ATIS Material: LDC CD-ROMs

In order to create Swedish ATIS material, it was decided to obtain the original American ATIS material (Hemphill *et al.*, 1990) for translation. Translations were considered the fastest and cheapest way to obtain data, if lacking from qualitative drawbacks. Consequently, CD-ROMs were purchased from the Linguistic Data Consortium (cf. References).

3.2.2 Swedish Material: The Stockholm-Umeå Corpus

In order to obtain Swedish text, the Stockholm–Umeå Corpus, or SUC for short (Ejerhed *et al.*, 1992; Källgren 1991; Källgren 1990), was consulted. The SUC corpus is relatively small, covering 1,000,000 words. Instead, within the SUC project, emphasis has been put on balance of material, and as a result thereof, several different literary disciplines are covered. Since the main interest of the SLT project was not the analysed corpus, but plain, raw text, unanalysed texts were obtained from the SUC material.

3.3 Text Material Used in Data Collection

The text material used in the SLT-2 project for the training of the Swedish speech recognizer falls into the following parts. NB: some of these parts were not included from the beginning of the project, but were included at later stages. However, it is pointed out when this is the case.

3.3.1 Practice Sentences

Each recording session commenced with a set of twelve practice sentences. These were recorded, but not used in the training. During the recording of these, the session leader could make comments about the recordings, and the speaker could ask questions if something was not clear.

3.3.2 Calibration Sentences

In order to get good acoustic coverage, all speakers recorded a set of fifty sentences that were phonetically and phonologically balanced, i.e., contained as much phonetic and phonological information about Swedish as possible. The average sentence length was nine words. When creating the calibration sentences, care was taken to cover the following:

1. Phonemic inventory.
2. Allophonic variation.
3. Formant transition coverage.

Concerning 1) above, traditional phonetic descriptions of Swedish standard phonemic inventory were consulted. The allophonic variation referred to in 2) mainly covers phonological processes like retroflexation of Swedish /n/, /t/, /d/, /l/ and /s/. The formant transitions mentioned in 3) accounts for the fact that vowel formant patterns vary according to the type of consonant that precedes or follows the vowel in question, something which have acoustical consequencies, and therefore is of interest for acoustic-phonetic training. Thus, a set of sentences were created where as many different formant transition patterns as possible were accounted for. This was achieved by creating a set of consonants with nine different places of articulation, *viz.*:

Bilabial	(p b m)
Labio-dental	(f v)
Dental plosive	(t d)
Dental sonorant	(n l)
Dental fricative	(s)
Retroflex	(r ʂ ʈ ɖ ɭ ɳ)
Palatal	(j)
Velar	(g k ŋ)
Uvular	(h)

A set of of “extreme” vowels was also created, namely:

Closed front	(i y)
Open back	(a)
Closed back	(o)

Since formant transitions occur both into and out of vowels, the sentences were checked according to how many of the total number of possible formant transitions were included. With nine consonant places of articulation and three vowels, this gives $9 \times 3 \times 9 = 243$ possible formant transition combinations. Of these, only 54 were lacking in the set of calibration sentences, i.e., 23 %. Moreover, the combinations not covered are rather rare, like palatal–open back vowel–palatal ("jaj"), and so on. As a matter of fact, it was found hard to find words with the missing patterns in the text material available in computer-readable format.

3.3.3 Calibration Sentences – ATIS

This set of sentences was added to the recording session material at a later stage of the project. The set contains fifty sentences with the hundred most common ATIS words, the sixty most common city names and the ten most common carrier names.

3.3.4 Calibration Sentences – Newspaper Texts

This set of sentences was added to the recording session material at a later stage of the project. This set contains forty sentences collected from the SUC corpus, slightly transformed for greater phonetic and phonological coverage. Of the 243 possible formant transitions patterns mentioned above, 53 are lacking, i.e., 22.6 %.

3.3.5 ATIS Sentences

This set of sentences contained the material described in the previous section. Thus, the bulk of the material consisted of translated American ATIS sentences, mainly from the Email Corpus (cf. LANGUAGE CHAPTER).

3.3.6 Expanded Phone Set

The American ATIS vocabulary contains certain words and names whose American pronunciation includes phonemes very different from the Swedish phonemic inventory. While English/American diphthongs (lacking in standard Swedish) might be approximated reasonably well by vowel+/j/ combinations, and some fricatives, like /ʃ/ are very similar to specific Swedish sounds, there are certain sounds that are too phonetically remote to be approximated by the normal Swedish phonemic/phonetic "toolbox". A number of such phonemes are shown in Figure 3.1.

However, Sweden is a country where a lot of people speak quite good English, and where it is to be expected that several speakers do possess the means to approximate English and/or American phones outside, and distinct from, the Swedish phone set. This could confuse training, since one could expect greater variability on such words. For example, the airline "Northwest", could be rendered both as [noːɽvɛst], [noːɔˑθwɛst] and so on. In order to account for this variability, a set of twelve sentences containing ten foreign – mainly English/American – names was added to the training material. Three examples of these sentences are:

Phoneme	Example Word
/θ/	Jethro, thriller
/ð/	the
/z/	Oz
/ʒ/	leisure
/ʧ/	Thatcher, Charles
/ʤ/	Gianna, James
/ɚ/	Price, Roger
/w/	plywood, White
/aː/	Aachen, Gonzales
/ʊə/	tour

Figure 3.1: Non-Swedish phonemes without close approximations in Swedish.

Veckopressens favoriter är verkligen Diana och Charles
Många har Roger Moore som favorit i rollen som James Bond
Den mest säljande skivan någonsin är "Thriller" av Michael Jackson

Other American–Swedish differences were judged to be small enough to be approximated by Swedish phones or combinations of Swedish phones. Phonemes that could be fairly well approximated by using standard Swedish phonemes are shown in Figure 3.2.

Original	Swedish	Example
/x/	/x, ɧ, ʂ/	Aachen
/ʃ/	/ʃ, ʂ/	Sharon
/ɬ/	/ɬ, l/	Elvis
/æ/	/æ, a/	Sharon, Jackson
/aʊ/	/aʊ/	aula, mouse
/eʊ/	/eʊ/	Europa
/eɪ/	/ei, ej/	Baywatch
/aɪ/	/ai, aj/	Michael
/ɔɪ/	/oi, oj/	Lloyd

Figure 3.2: Non-Swedish phonemes with reasonable Swedish approximations.

For an analysis of this material and a fuller discussion of Swedes' competence of English/American pronunciation in Swedish sentences, cf. Eklund and Lindström (1996).

This material has, so far, not been used for training.

3.4 Dialect Areas

In selecting the areas to cover when collecting dialect data, there are basically two problems to consider, one linguistic and one practical.

The linguistic problem concerns the definitions of what can be considered a “dialect”. Swedish dialect research has a long tradition, represented by e.g. Meyer (1937; 1954), Gårding (1975), Bruce and Gårding (1978), Elert (1994), Fries (1994) and others. These, and other, sources were consulted in order to establish the relevant linguistic parameters to consider.

The practical reason concerns finding places where recordings actually could be carried out. For this reason, only places where Telia branches are located were used. This meant that rooms for the recordings could be found, as well as Telia employees who were willing to help with the recruitment of local subjects, or even constitute part of the group of subjects themselves, provided they were “local sons or daughters”.

Thus, in some kind of compromise between linguistic desiderata and practical constraints, some fifty recording sites were chosen, covering most of Sweden, rather more following population density than geographical spread.

3.4.1 Subjects

A pre-requisite for successful, general speech recognition, is that the training material is well balanced. Therefore, it is of vital importance that the people recorded be representative as to age spread, gender, social strata and so forth. It was decided to record people between 15 and 65 years of age, and to try to keep an equal number of men and women. Consequently, for the Stockholm recordings, a careful balance was maintained over sex and over ages from 15–65 (in five-year age groups). However, for the dialect recordings, an equal number of male and female subjects were recorded, but due to “recruiting” problems, the balance over ages was only maintained to the extent that it was possible.

3.5 Recording Procedures

The recordings were done with parallel recording of the speech from two microphones, one close-talking high-quality microphone attached to a headset, and one telephone handset.

3.5.1 Computer

The recording machine was a Sun Sparc 5, and a standard 16-bit A/D-converter was used. The speech data was stored on a 9 Gbyte disk. A DAT tape station was used for nightly backups.

3.5.2 Headset

The subjects used a Sennheiser HMD 410-6 headset with a close-talking microphone. This had the advantage that one of the ear-pads could be swiveled to the side, leaving one ear free.

3.5.3 Telephones

Ten different types of telephones were used, listed in Table 3.3. The normal telephone were fed with 50 Volts via a bridge, and the signal was picked up from the line output. The cellular phones were essentially inactive, except that the microphone circuitry was fed from a 4.5 Volt battery. The signal was picked up from the output of the microphone preamplifier.

Doro Gamma
Doro Diplomat
Doro Rapport
Ericsson OH 198
Ericsson OH 337
Häger Athena
Target MX 801
Target HF-20
Telia Bizz 55
Telia Diavox 1
Telia Dialog
Telia Gazette
Telia Ouno
Telia Respons
Zodiac Sigma
Zodiac Sigma 300
Zodiac Sigma 500

Figure 3.3: List of telephones used in the speech data collection.

The rationale for doing this is that signals recorded in this way can be fed back into a telephone line from a source with a 600 Ohm impedance, and essentially look as if they came directly from the original device. The difference would be a pair of A/D and D/A conversions, but the effect of these is hopefully negligible, compared to the distortions in the telephone network.

The signal from the cellular phones can likewise be fed back into the phones at the point where the preamplifier would have been connected. The signal from the cellular phone can then be passed to test bed, where different carrier/interference ratios and and fading conditions can be imposed.

Others (Brown and George 1995) have used artificial heads to reuse high-quality recordings to generate telephone quality speech, but we hope that our procedure will affect the signal much less than an artificial head.

3.5.4 The SRI Generic Recording Tool

The SRI Generic Recording Tool (GRT) was used to present the text to be spoken to the subjects and to control the recording device.

The subjects controlled the recording themselves by using mouse buttons and a graphical user interface. They could move forward and backward through the recording scripts. They could also play-back their recordings to verify the results.

The recordings were divided into several sessions, listed in Section 3.3. For the sessions which consisted of reading general (newspaper) texts, entire paragraphs were presented, with the sentence to be read was in inverse video.

Each sentences was stored in a separate file by the GRT, together with a file containing the prompt text.

3.5.5 Settings

Two adjacent rooms were used, one for the subject and the Sparc 5 terminal, the other for the recording equipment and the recording supervisor. Some attempts were made to control the acoustic damping of the subject's room, but for the dialect recordings, it was frequently necessary to simply use the locations that were available.

3.6 Checking

The correctness of the recordings was checked by listening to the speech files. This was aided by a program which played a file, and then asked the verifier to either confirm the correctness, change the text to correspond to the actual utterance, or to mark truncations or strange words.

The following classifications were used:

OK A good recording.

FOREIGN_WORD Good, but contains some foreign word.

BAD_LEXEM Good, but contains some non-lexical word.

DISFLUENCY Some word is spoken in a disfluent manner, but most of the utterance is good speech.

TRUNCATED Part of the utterance is truncated, but the rest is good speech.

REPETITION Contains a restart and repetition. This could be useful speech data, but was considered not worth the effort to transcribe.

NOISE Bad, don't use.

BAD Really bad. Complete silence, just noise, or no good speech left.

Disfluencies and foreign words were marked by surrounding them with asterisks. Truncated parts of a prompt sentence was replaced by opening and closing square brackets. When the speech recognizer is trained on this data, anything within asterisks or square brackets is replaced with rejection models.

Note that the tag **NOISE** marks noise within the speech. Noise before or after the utterance has been ignored, assuming that some 'silence' model will handle heavy aspiration and similar phenomena.

3.7 Lexicon

The recognizer works against a lexicon, and consequently a lexicon containing the vocabulary to be recognized needed to be created.

An issue to consider is to what extent alternative pronunciations should be catered for. The way people pronounce words do not only differ inter-individually, but also intra-individually. The same person may use different degrees of reduction in his/her speech, and these may vary from time to time. Moreover, some reductions are more likely to occur than other, and some reductions have a tendency to co-occur. The main issue here is to decide exactly how reduced the included forms should be, and how much should be left to training proper. Within the SLT project, a method where most things were left to the recognizer was opted for, which meant that heavy reductions were not included. On the average, only some very common reduced forms were considered, notably grammatical endings, since these are almost invariably reduced in speech.

In order to create such a lexicon, a base lexicon was first created in order to provide 'standard' pronunciation, i.e., clear standard Swedish. The lexicon was not phonological, but rather represents clearly articulated pronunciation. An excerpt from the 'base' lexicon looks thus:

```
egentligen          e j e n t l i g e n
ekipage             e k i p a: rs
ekorrar             e k aa r a r
eller               e l e r
elvahundra          e l v a h u n d r a
emotionell          e m o t sj o n e l
```

The phonetic symbols used were inherited from previous systems. Each phone is denoted by one or more characters and is separated by a space. A colon indicates a long vowel, otherwise the vowels is short.

In order to cover alternative, more reduced pronunciations, a set of phonological rules were written that automatically generated alternative surface forms. The rules were conceived as phonological rewrite rules, of 'standard' format:

$$\alpha \longrightarrow \beta \;/\; \Gamma \_ \Sigma$$

... which reads: α becomes β between Γ and Σ.

A couple of these rules are shown in Figure 3.4.

```
%==========
Rule block: /igt/-regler

Rule: ‘‘/g/ avtonas mellan kort /i/ och /t/. Ex: ’gruvligt’ -> ’gruvlikt’’’
syll:[-primary] phn:[‘‘g’’] --> phn:[-voice] / phn:[‘‘i2’’] _ phn:[’’t’’]

%==========
Rule block: /ikt/-regler

Rule: ‘‘/k/ stryks mellan kort /i/ och /t/. Ex: ’gruvlikt’ -> ’gruvlit’’’
syll:[-primary] phn:[‘‘k’’] --> phn:0 / phn:[‘‘i2’’] _ phn:[’’t’’][+wb]
```

Figure 3.4: Example of phonological rewrite rules

The new forms were added to the lexicon immediately after the base forms, thus:

```
egentligen          e j e n t l i g e n
egentligen          e j e n t l i j e n
egentligen          e j e n k l i j e n
ekipage             e k i p a: rs
ekipage             e k i p a: sj
ekorrar             e k aa r a r
eller               e l e r
elvahundra          e l v a h u n d r a
elvahundra          e l v a h u n r a
emotionell          e m o t sj o n e l
emotionell          e m o sj o n e l
```

The lexicon thus created contains all words from the calibration and newspaper texts.

For the ATIS material, several lexica were produced. There are three reasons for this:

- It is important from a performance point of view, to have an accurate lexicon for the task domain, and since the vocabulary is comparatively small, it is also doable. Special lexica was created for the ATIS sentences, which are more strictly pruned than the main lexicon. All lexical ambiguities have been removed, so that only word senses and pronunciations that belong in the ATIS domain are left.

- During training, it is necessary to allow all possible pronunciations of the words in the vocabulary, but during recognition it is better to canonicalize the pronunciation in a way that makes sense for the language component. In the ATIS domain, the main differences are in the pronouns “de”, “dem”, “dom”, “dig” and “sig”. These are on the other hand very common.

- Swedish is language with very productive compounding, which increases the vocabulary size, compared to English. Also, natural numbers and monetary amounts are compounds, and these are, for obvious reasons, impossible to list in a lexicon (at least in the general case; in ATIS, there probably is an upper limit to the cost of an air fare). In all ATIS lexica, natural numbers and monetary amounts have been split into their components. For the other compounds, split and unsplit versions of the lexicons have been produced to allow experimenting. The term "split" means that all compounds have been split at word boundaries, but affixes have been retained.

3.8 Concluding Remarks

Summing up the speech collection activities within SLT, there are many things to comment upon.

First, a general thing to point out is that the work was in very many ways hampered by practical considerations, and thus "constrained" in undesirable ways. Some of these are listed below.

Recording Environments Much of the recordings were carried out under less-than-perfect circumstances. However, although the rooms in which the recordings took place *could* have been dampened to obtain better acoustics, this would have taken a lot of time and required a lot of effort which very much would have delayed the data collection. One simply had to accept the trade-off between recording quality and time allotted.

Morphological Information Since morphological features were not available for the lexicon entries, some overgeneration of the reduction rules was the result, and manual pruning was required.

WOZ data Data collected from read speech naturally is flawed, not only from a "language" point of view (cf Chapter 2), but also from a speech point of view. People often find it hard to read prompts from a computer screen in a fluent and natural way, and at the same time handle the recording tool, executing mouse clicks and so forth.

Dialect Handling in the Lexicon The present lexicon does not handle dialects where retroflexing of dental consonants does not occur (see Chapter 4).

However, despite all those desiderata and wishes not fulfilled or realised during this stage of the project, the results are overwhelmingly positive. Things like word error rate – even after the first training session – speak for themselves!

Chapter 4

Speech Recognition

Harry Bratt, Vassilios Digalakis, Robert Eklund, Horacio Franco, Jaan Kaja, Leonardo Neumeyer, Patti Price and Fuliang Weng

The major engineering and research accomplishments of the speech recognition part of the SLT-2 project have been:

- Design, specification and collection of a Swedish speech corpus for training speech recognition models, support dialect adaptation research, and over-the-telephone speech recognition research (Section 4.1).
- Design and generation of a Swedish pronunciation dictionary for speech recognition (Section 4.2).
- Investigation, and implementation of medium-size vocabulary, speaker-independent, continuous, real-time, accurate, English and Swedish speech recognition systems (Section 4.3).
- Investigation of acoustic dialect-adaptation techniques: solving the problem of training Hidden Markov Models (HMMs) with limited amounts of acoustic data (Section 4.4).
- Investigation of language-modeling techniques: solving the problem of training backoff language models (LMs) with limited amounts of orthographic data (Section 4.5).
- Development of a Swedish/English aligner for one-best and N-best applications. Development of a phone backtrace for the N-best decoding algorithm (Section 4.6).
- Investigation and development of a multilingual speech recognition system (Section 4.7).

4.1 The Swedish Speech Corpus

Leo and Jaan to write.

4.2 The Swedish Lexicon

4.2.1 Introduction

Our starting point for the Swedish lexicon was a lexicon originally designed for a speech synthesis task. The synthesis lexicon contained a larger number of phones than are useful in recognition system; phones which are very infrequent could still be synthesized, but could not be modeled well for a recognition task since there would not be enough data.

4.2.2 Phone Set

Selecting a phone set is a crucial first step in creating a new recognition lexicon. For Swedish, that selection posed two main challenges – phones from loan words, and the retroflex series of phones introduced by phonological rules.

The training corpora (see Section 4.1 above) contained a small number of loan words for which the initial lexicon had pronunciations containing non-Swedish phones, such as the voiceless velar fricative from German or the voiced palatal fricative of French. For the purposes of the ATIS system, these words could be classified into two groups: those within the ATIS corpus and those within other training corpora only. For the phones which only occurred in non-ATIS words, we simply eliminated the utterances containing those words from the training set – they comprised a very small number. The phones which occurred in ATIS words, we mapped to native Swedish phones. These were mainly English diphthongs which were mapped to a sequence of two Swedish vowels.

The problem in dealing with the retroflex series was an instance of the general problem of what level of representation to use for any speech recognition lexicon, on the spectrum of phonetic to phonological. The retroflex phones in Swedish are not separate phonemes, but rather are allophones derivable from underlying coronal phonemes when preceded by /r/. The domain of this rule is larger than the word level, so that a word beginning in, e.g., an /n/ occurring after a word ending in an /r/ may become retroflex. We considered three ways of representing this in the lexicon. First, we considered leaving the retroflexes out completely and representing only the underlying forms of the words in the lexicon. Thus, we would essentially rely on triphones to model sequences such as /r l/ as though there were really one phone "/rl/" there. The other two possibilities involved representing the retroflex phones in the lexicon. When dealing with a word with an underlying non-initial /r l/ sequence we would represent that with the single phone /rl/. There is still a question, however, of how to deal with words beginning with a coronal – since these coronals are susceptible to undergoing the retroflex rule if they follow a word ending in /r/. Thus, the second representation we considered involved making multiple pronunciations for all coronal-initial words

(one starting with the non-retroflex allophone, the other with the retroflex allophone), whereas the third possible representation involved ignoring the word-initial variations, and representing the retroflex process only as a lexical process (i.e. within words in the lexicon, not across words). The first representation was impractical because we could not easily recover the underlying forms from the lexicon we were starting with. We decided on the second representation over the third in hopes of capturing some of the cross-word changes that would occur. The possible disadvantages of this approach are, first, that we overgenerate pronunciations, since we're not constraining where the retroflex pronunciation may occur (i.e. only after words ending in /r/), and second, that the true rule would not only retroflex the initial coronal, but would delete the previous /r/ as well, and we are not modeling that.

4.2.3 Morphology

In previous recognition systems for English we have not dealt with the issue of morphology. That is, to build a vocabulary and language model, we simply took words from a training corpus in the inflectional forms in which they appeared. Swedish, however, has a richer system of inflectional morphology than English, so a word may appear in many different forms. Some of those forms might not exist in a training set, but could be predicted to occur in a test set. An example of this in English might be where the word "flights" appears in a training set but not the singular form "flight." It could be reasonable to conclude that the singular form would occur in a test set.

We considered three different methods of handling morphology in the system. The simplest method is to collect all words seen in the training corpora, and generate the language model on exactly what was seen, in the same way we handled the English system. This ignores the potential problem that other inflectional forms of words which were seen in the training set may exist in the test set. If a language has a large number of inflectional forms of any given word, we might expect that there would be significant gaps in the training data which would be predictable from the morphology. Though this is not a problem for English, we were concerned that it could be a problem for Swedish. A second method would be to try to account for unseen forms in the lexicon based on the observed forms in the training set. A large amount of linguistic knowledge would be needed here, not only to account for all forms, both regular and irregular, but also to attempt to account for the distribution of the unseen forms in the language model – that is, how do you statistically predict the unseen form of the word given some context. Finally, a third method would involve treating not a word, but each morpheme, as the lexical item processed by the recognizer. For example, in English, "flights" would be treated as two lexical items: "flight" and "s." This method may be a good idea for languages with a lot of concatenative morphology – though it is almost certainly unwarranted for a language like English which has very little productive morphology (at least, inflectional morphology, which is the most relevant type here).

The third approach, in fact, also seemed unwarranted for Swedish. We considered trying the second approach, but decided it was too formidable for a baseline system, since it would involve a significant amount of research and design to create all of a word's useful inflectional forms and to incorporate the unseen ones into the statistical language model which we are using. Therefore, we took the straightforward approach

of modeling only those forms which were seen in the training data – the same approach as we have taken for English.

4.2.4 Lexicon Statistics

The final Swedish ATIS lexicon contained 1265 words. The average number of pronunciations per words was 1.68. In comparison, the previous English lexicon for ATIS contained 1751 words, with an average of 1.41 pronunciations per word. If we had not included the alternate retroflex pronunciation of all coronal-initial words (see Section 4.2.2), the number of pronunciations per word in the Swedish lexicon would have gone down to 1.22.

4.3 The English/Swedish Speech Recognition System

Two of the major goals of the SLT-2 project were to improve the speed and accuracy of the speech recognition engine in the English ATIS domain and for the first time to build a Swedish ATIS recognizer. To achieve these goals we decided to use the Nuance speech recognition engine, which is a productized version of the Decipher engine. We extended the Nuance engine to support the statistical language models required for the ATIS domain. The current Nuance engine only supports finite-state grammars. The ATIS English and Swedish models were trained by SRI and optimized to provide maximum accuracy at real-time performance on an UltraSparc computer platform. The major difference between the SLT-1 and SLT-2 systems is that the latter uses continuous density hidden Markov models (HMMs). We investigated the use of two different approaches: phonetically-tied mixture systems (PTM) and genonic systems (Digalakis, Monaco and Murveit, 1996).

- Phonetically-tied mixture system (PTM) (Paul, 1989; Lee *et al.*, 1991; Aubert *et al.*, 1993). This is a phone-based tying continuous density HMM. Only HMM states that belong to allophones of the same phone share the same mixture components.
- Genonic systems (Digalakis, Monaco and Murveit, 1996). The SRI genonic system is used to achieve a good trade-off between modeling resolution and robustness. It uses a general scheme for tying mixture components in continuous mixture-density HMM-based speech recognizers. The sets of HMM states that share the same mixture components are determined automatically using agglomerative clustering techniques.

4.3.1 Diagnostic Experiments

We first ran some baseline experiments to compare the various speech recognition systems without optimizing for speed. We compared the old vector quantization (VQ) system used in the SLT-1 project with various genonic and PTM systems. We used the same task as in the SLT-1 project, that is a ten-city ATIS grammar. Experiments were carried out using SRI's DECIPHER^{TM} speech recognition system configured with a

HMM Type	Num gen	Num Gauss gen	Total num Gauss	Ins (%)	Del (%)	Sub (%)	WER (%)	x cpuRT (Sparc20)
GEN	600	32	19K	0.4	1.6	4.9	6.8	18.2
GEN	1100	32	35K	1.1	0.7	5.3	7.0	26.4
GEN	390	32	12K	1.5	0.7	5.3	7.4	14.3
PTM	38	100	4K	1.0	1.3	5.4	7.7	9.9
GEN	600	16	10K	1.9	0.9	5.4	8.3	12.9
VQ	-	-	-	0.3	3.2	7.5	11.0	5.5

Table 4.1: VQ, PTM, and Genonic word error rates on a 10-city English ATIS task.

HMM Type	Ins (%)	Del (%)	Sub (%)	WER (%)
ATIS	0.5	1.8	5.4	7.7
WSJ	1.5	1.2	8.0	10.6

Table 4.2: Word error rates on a 46-city English ATIS task. HMMs are trained using ATIS or WSJ acoustic data.

six-feature front end that outputs 12 cepstral coefficients ($c_1 - c_{12}$), cepstral energy (c_0), and their first- and second-order differences. The cepstral features are computed from an FFT filterbank, and subsequent cepstral-mean normalization on a sentence basis is performed. To train the models we used 21,000 ATIS sentences. A summary of these experiments is shown in Table4.1.

The results show a reduction of 40% in word error rate (from 11.0% to 6.8%) for the best genonic system compared to the SLT-1 VQ system. We also observe a significant increase in computational complexity (cpu real-time for VQ is 5.5 compared to 18.2 for the best genonic system). Our goal for the SLT-2 project is to maintain the increased accuracy provided by the continuous density HMM while increasing the speed so we can achieve real-time performance. We also observe that a PTM system may be more appropriate for this task, given that the increase in accuracy over the VQ system is still significant while this system is much faster than the genonic ones.

Another diagnostic experiment is summarized Table 4.2. In this test we study the relevance of using domain-specific acoustic data for training the HMMs. We trained a "Wall Street Journal" (WSJ) system and an ATIS system using comparable amounts of training data. For testing we used a 46-city ATIS test set and identical backoff bigram language models. The results show that using ATIS-specific training data can result in significant improvement compared to using only WSJ data. (WER 7.7%, as compared to 10.6% for WSJ). This result had direct implications for the design of the data collection specification.

4.3.2 Speed Optimization

To achieve real-time performance various speed optimization techniques were investigated:

Viterbi Beam Search Pruning This technique is used in the standard Viterbi beam search algorithm. The pruning is implemented at the phone level: all three HMM states for a given allophone are either pruned or kept. The Viterbi pruning can be modified from external configuration files or from the command line. The parameter name is "Pruning" and typical values range from 800 for aggressive pruning to 1200 for little pruning.

Model-Specific HMM Update Routines Code is highly optimized for specific HMM topologies.

Skip Frames Exact (SkipE) HMM updates are computed only every two frames instead of every frame.

Gaussian Pruning This technique aborts Gaussian computation if the probability, up to a given dimension in the vector, is below a given threshold.

Skip Observation Frames (SkipObsFrames) Depending on the value of the SkipObsFrame parameter, certain hypotheses in the search will be approximated by repeating Gaussian values from the previous frame. When the SkipObsFrames parameter is zero, all hypotheses use the approximation. As the value increases there is less approximation.

Shortlists (Digalakis, Monaco and Murveit, 1996). This technique significantly reduces the amount of Gaussian computation. Gaussian shortlists are lists that specify the subset of the Gaussians distributions expected to have high likelihood values in a given region of the acoustic space.

Phonetic Pruning (PPR) This technique is used to prune out hypotheses using phones whose probabilities are below a certain threshold.

To optimize the system we varied the parameters sequentially to determine the optimum set. A summary of the optimization runs for the English PTM system is shown in Table 4.3. The PTM system is gender-independent, uses 56 classes and 100 Gaussians per mixture. The test set consist of 200 waveforms and 2000 words. All tests were run on a Sparc20-50Mhz machine. The table shows the results for four techniques: Viterbi pruning (PRUNE), Gaussian pruning (GPRUNE), shortlists (SHORTL), and SkipObsFrames (SOF). We see that aggressive Viterbi pruning can significantly reduce computation but at the cost of high error rates. For example, reducing the cpu time from 11.3 to 5.8 times real time results in an increase in word-error rate from 7.8% to 9.6%. This is due to the fact that we are pruning correct hypotheses during the search. In general, the Viterbi pruning threshold can be set to 1000 without sacrificing accuracy.

Based on the first round of optimization, we select the values that result in maximum speed-up with little degradation in recognition accuracy. The next step consist in optimizing the phonetic pruning parameters. This optimization further reduces the cpu

Baseline run (No optimization)								
PRUNE	GPRUNE	SHORTL	SOF	INS	DEL	SUB	WERR	x cpuRT
1200	75K	no	100K	1.6	1.7	4.6	7.8	11.3
Shortlists (values from 1.0 down to 0.7)								
PRUNE	GPRUNE	SHORTL	SOF	INS	DEL	SUB	WERR	x cpuRT
1200	75K	1.0	100K	1.5	1.6	4.5	7.7	7.6
1200	75K	0.975	100K	1.5	1.7	4.5	7.6	6.9
1200	75K	0.950	100K	1.4	1.8	4.6	7.8	7.1
1200	75K	0.925	100K	1.4	1.9	4.4	7.6	7.5
1200	75K	0.90	100K	1.3	1.8	4.5	7.5	7.3
1200	75K	0.85	100K	1.4	1.6	4.3	7.3	7.0
1200	75K	0.80	100K	1.4	1.6	4.8	7.7	7.0
1200	75K	0.70	100K	1.5	1.5	4.9	7.9	5.9
Viterbi Pruning (values from 1000 down to 600)								
PRUNE	GPRUNE	SHORTL	SOF	INS	DEL	SUB	WERR	x cpuRT
1000	75K	no	100K	1.6	1.7	4.7	7.9	8.5
900	75K	no	100K	2.0	1.7	5.0	8.6	6.9
800	75K	no	100K	2.4	1.8	5.5	9.6	5.8
700	75K	no	100K	3.0	1.8	7.1	11.9	4.8
600	75K	no	100K	4.6	1.7	10.2	16.5	3.7
Gaussian Pruning (values from 50000 down to 4000)								
PRUNE	GPRUNE	SHORTL	SOF	INS	DEL	SUB	WERR	x cpuRT
1200	50K	no	100K	1.6	1.7	4.5	7.7	10.9
1200	30K	no	100K	1.6	1.6	4.4	7.6	9.1
1200	20K	no	100K	1.6	1.7	4.2	7.4	9.0
1200	10K	no	100K	1.8	1.6	4.4	7.8	8.1
1200	8K	no	100K	1.9	1.6	4.7	8.0	7.3
1200	6K	no	100K	1.9	1.5	4.4	7.6	7.3
Skip Obs Frames (values from 1000 down to 0)								
PRUNE	GPRUNE	SHORTL	SOF	INS	DEL	SUB	WERR	x cpuRT
1200	75K	no	1000	1.6	1.7	4.6	7.8	11.5
1200	75K	no	800	1.5	1.7	4.5	7.7	10.1
1200	75K	no	600	1.6	1.8	4.5	7.8	10.0
1200	75K	no	400	1.6	1.8	4.6	8.0	8.4
1200	75K	no	200	1.6	1.8	4.4	7.7	7.5
1200	75K	no	0	1.7	1.7	4.4	7.8	7.6

Table 4.3: Optimization of the English ATIS PTM system.

SWE	ENG	SWE	ENG
a	ah	o	aa
a:	ah	oe	axr
aa	aa	oe2	axr
aa:	aa	oe2:	axr
ae	ae	oe:	axr
ae2	ae	ow	ow
ae2:	ae	p	p
ae:	ae	r	r
ay	ay	rd	d
b	b	rl	l
d	d	rn	n
e	eh	rs	sh
e:	ey	rt	t
f	f	s	s
g	g	sh	sh
h	hh	sj	sh
i	ih	t	t
i:	iy	th	th
j	y	tj	ch
k	k	u	uh
l	l	u:	uw
m	m	v	v
n	n	w	w
ng	ng	y	y
o:	aa	y:	iy

Table 4.4: Mapping phonemes from English To Swedish for initialization.

requirements to 1.5 times real-time on the Sparc 20 host with an error rate of 8.8%. The exact operating point can easily be adjusted based on the cpu cycles available.

4.3.3 Swedish Recognition

Our approach for building a Swedish recognizer was to boot the training process using English models. To do this, we created an approximate mapping from English to Swedish phonemes (see Table 4.4).

We trained Swedish gender-independent PTM models using 23,000 utterances from 94 speakers. In Table 4.5 we show English and Swedish baseline recognition results before any speed optimization is carried out on small 200-sentence test sets. We notice that the main differences between the Swedish and English experiments are the number of training speakers and out-of-vocabulary (OOV) rate. A larger number of training speakers makes the recognizer more robust to variation in speaker voices. The larger

Language	Swedish	English
Training speakers	94	408
Training sentences	23K	21K
Test speakers	27	29
Test sentences	200	200
Test words	1749	2000
Perplexity	24	20
Out-of-vocab words	2.2	0.1
Insertion errors	1.8	1.6
Deletion errors	1.2	1.7
Substitution errors	6.0	4.6
Word error rate	9.2	7.8

Table 4.5: Comparison of English and Swedish baseline recognition experiments.

OOV rate in the Swedish test set is probably producing the difference in error rate comapred to the English system. Using a conservative estimate of one OOV resulting in one error, we could assume that the adjusted Swedish error rate is in the order of (9.2% - 2.2% = 7.0%), that is, similar to the English word error rate.

4.3.4 Summary

We implemented real-time English and Swedish speech recognition systems for the ATIS task. Based on the available test sets, it appears that the speech recognition performance (after adjusting for errors caused by out-of-vocabulary words) is similar for both languages. The operating point in the speed-accuracy curve can be adjusted using various optimization techniques achieving real-time performance on the target platform.

4.4 Dialect Adaptation

We are interested in developing speech recognizers that are robust to the large dialect variability that exists in spoken Swedish. However, the recognition accuracy of large-vocabulary speech recognition systems has proven to be highly related to the correlation of the training and testing conditions. Performance degrades dramatically if a mismatch exists between these conditions, such as different channel, speaker's voice characteristics, or, in our case, dialect.

In this work, we consider the dialect issue on a speaker-independent (SI) speech recognition system. Based on the Swedish language corpus collected by Telia, we investigate the development of a Swedish multi-dialect SI speech recognition system which will require only a small amount of dialect-dependent data. We first investigate the effect of mismatched conditions in training and testing, and we find that the

recognition performance of a speaker-independent system trained on a large amount of training data from the Stockholm dialect decreases dramatically when tested on speakers of another Swedish dialect, namely from the Scania region.

To improve the performance of the SI system for speakers of dialects for which minimal amounts of training data are available, we use *dialect adaptation* techniques. We apply both maximum likelihood (ML) transformation based approaches (Digalakis *et al.*, 1995; Neumeyer *et al.*, 1995; Legetter and Woodland, 1995; Sankar and Lee, 1996), as well as combined transformation-Bayesian approaches (Digalakis and Neumeyer, 1996), in an effort to minimize the effect of different dialects.

4.4.1 Dialect Adaptation Methods

The SI speech recognition system for a specific dialect is modeled with continuous mixture-density hidden Markov models (HMMs) that use a large number of Gaussian mixtures (Digalakis *et al*, 1996). The component mixtures of each Gaussian codebook (*genone*) are shared across clusters of HMM states, and hence the observation densities of the vector process y_t have the form:

$$P_{SI}(y_t|s_t) = \sum_{i=1}^{N_\omega} p(\omega_i|s_t) N(y_t; m_{ig}, S_{ig}),$$

where g is the genone index used by the HMM state s_t.

These models need large amounts of training data for robust estimation of their parameters. Since the amount of available training data for some dialects of our database is small, the development of dialect-specific SI models is not a robust solution. Alternatively, an initial SI recognition system trained on some *seed* dialects can be adapted to match a specific *target* dialect, in which case the adapted system utilizes knowledge obtained from the seed dialects. We choose to apply algorithms that we have previously developed and applied to the problem of speaker adaptation, since in our problem there are consistent differences in the pronunciation between the different dialects that we examine. The adaptation process is performed by jointly transforming all the Gaussians of each genone, and by combining transformation and Bayesian techniques.

The transformation part of the adaptation process can be simply described by writing the observation densities of the dialect-adapted (DA) models as:

$$P_{DA}(y_t|s_t) = \sum_{i=1}^{N_\omega} p(\omega_i|s_t) N(y_t; A_g m_{ig} + b_g, A_g S_{ig} A_g^t). \tag{4.1}$$

To adapt the initial SI recognition system, the parameters $A_g, b_g, g = 1, \ldots, N_g$ have to be estimated. N_g denotes the number of transformations for the whole set of genones. The parameter estimation process is performed using the EM algorithm (Dempster *et al*, 1977). In our experiments we consider two variations of the generic transformation above. In the first variation (method I), we assume the matrix A_g is diagonal (Digalakis *et al.*, 1995), and is applied to both the means and covariances of the models, as in equation (4.1).

The second method (method II, in Leggetter and Woodland (1995) and Neumeyer *et al.*, 1995) assumes that A_g is a block diagonal matrix which transforms only the means of the Gaussian distributions:

$$P_{DA}(y_t|s_t) = \sum_{i=1}^{N_\omega} p(\omega_i|s_t) N(y_t; A_g m_{ig} + b_g, S_{ig}). \tag{4.2}$$

Each of the three blocks of this matrix performs a separate transformation to every basic feature vector (cepstrum, and its first and second derivatives). For the speaker adaptation problem, it was shown in Neumeyer *et al.*, (1995) that method II with a block diagonal matrix significantly outperform both method II with a full matrix and method I with a diagonal matrix.

Bayesian techniques use prior knowledge together with the small amount of training data to adapt the system. These techniques have several useful properties, such as asymptotic convergence and text independence. However, they suffer from slow adaptation rates. By combining the Bayesian with the transformation based approach, we expect to achieve faster adaptation as well as better convergence to the dialect-specific models as the number of training sentences increases. In order to implement the combined approach, we first adapt the SI models to match the new dialect using a transformation method. Then, these dialect adapted models serve as prior knowledge for the Bayesian adaptation step. For a more detailed description of how the combination is performed, the reader is referred to Digalakis and Neumeyer (1996).

4.4.2 Experimental Results

The adaptation experiments were carried out using a multi-dialect Swedish speech database collected by Telia. The core of the database was recorded in Stockholm using more than 100 speakers. Several other dialects are currently being recorded across Sweden. The corpus consists of subjects reading various prompts organized in sections. The sections include a set of phonetically balanced common sentences for all the speakers, a set of sentences translated from the English Air Travel Information System (ATIS) domain, and a set of newspaper sentences.

For our dialect adaptation experiments we used data from the Stockholm and Scanian dialects, that were, respectively, the seed and target dialects. The Scanian dialect was chosen for the initial experiments because it is one of three that are clearly different from the Stockholm dialect. The main differences between the dialects is that the long (tense) vowels become diphthongs in the Scanian dialect, and that the usual supra-dental /r/-sound becomes uvular. In the Stockholm dialect, a combination of /r/ with one of the dental consonants /n/, /d/, /t/, /s/ or /l/, results in supradentalization of these consonants and a deletion of the /r/. In the Scanian dialect, since the /r/-sound is different, this does not happen. There are also prosodic differences.

In addition, the Scanian dialect can be divided into 4 distinct areas (subdialects), namely Malmö, Helsingborg, Trelleborg and Kristianstad. In our experiments, the training and test sets consist of sentences chosen equally from the above subdialects in order to create a generic, subdialect-independent system. There is a total of 40 speakers of the Scanian dialect, both male and female, and each of them recorded more

than 40 sentences. We selected 8 of the speakers (half of them male) to serve as testing data, and the rest composed the adaptation/training data with a total of 3814 sentences. Experiments were carried out using SRI's $DECIPHER^{TM}$ system (Digalakis and Neumeyer, 1996).

The system's front-end was configured to output 12 cepstral coefficients, cepstral energy and their first and second derivatives. The cepstral features are computed with a fast Fourier transform (FFT) filterbank and subsequent cepstral-mean normalization on a sentence basis is performed. We used genonic HMM's with arbitrary degree of Gaussian sharing across different HMM states (Digalakis *et al.*, 1996b).

The SI continuous HMM system, which served as seed models for our adaptation scheme, was trained on approximately 21000 sentences of Stockholm dialect. The recognizer is configured so that it runs in real time on a Sun Sparc Ultra-1 workstation. The system's recognition performance on an air travel information task similar to the English ATIS one was benchmarked at a 8.9% word-error rate using a bigram language model when tested on Stockholm speakers. On the other hand, its performance degraded significantly when tested on the Scanian-dialect testing set, reaching a word-error rate of 25.08%. The degradation in performance was uniform across the various speakers in the test set (see Table 4.6), suggesting that there may be consistent differences across the two dialects. Hence, there is a great potential for improvement through dialect adaptation.

In the first set of experiments, we adapted the Stockholm-dialect system using various amounts of adaptation data from the training speakers of the Scanian dialect, and evaluated the performance of the adapted system to a separate set of testing speakers. This gives us a measure of the dialect-adapted, speaker-independent performance, since the adaptation and testing sets consist of different speakers. We also trained from scratch a Scania-dialect system using standard ML training based on the same adaptation data (ML-trained system), in order to estimate the adaptation benefits.

The results are summarized in Figure 4.1. We see that even with the first simplified algorithm, which does not take full advantage of large amounts of training data, we get a significant improvement in the performance. With as few as 198 sentences we get a 38% reduction and the word-error rate drops to almost 15%. Method II produces even better results, and the error rate for the same amount of training sentences falls to approximately 13%. However, when compared with the ML-trained system, we see that the transformation adaptation methods outperform the standard ML training only when a very small amount of training data is used (i.e. less than 400). For larger amounts of training data, the ML-trained system performs better, and this is due to the bad asymptotic properties of the transformation adaptation, as well as the relatively small vocabulary of the ATIS system.

In Figure 4.1, we also present the results of the combination of methods I and II with Bayesian adaptation. The combined schemes are proven to be far more efficient than the simple transformation methods I and II, and the adaptation takes better advantage of the amount of the training sentences. The error rate is reduced by 63%, 69% and 75%, with 198, 500 and 2000 adaptation sentences, respectively. Although no direct comparison can be made, using as few as 198 adaptation sentences, the error rate of 9.37% approaches the Stockholm dialect dependent performance. For more sentences the error rate drops even more, to 6.40%. In addition, the combined approach signif-

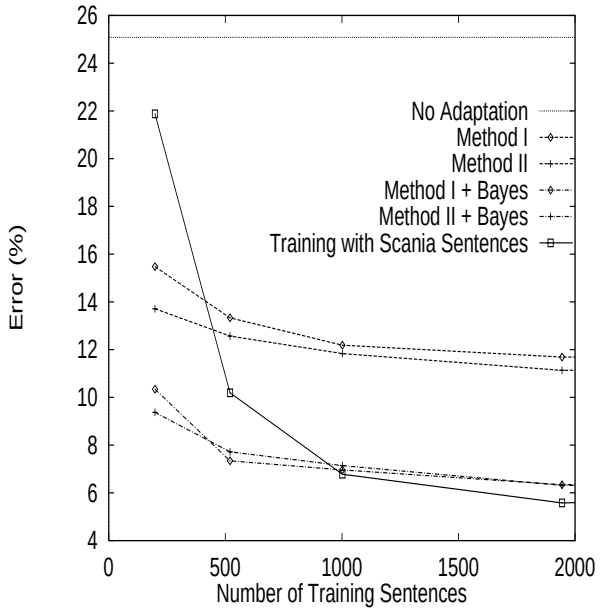


Figure 4.1: Dialect adaptation results for adaptation methods I, II, their combination with Bayes and standard ML training.

icantly outperforms the ML trained system when less than 1000 sentences are used, providing a solution that is more robust and easier to train.

In Table 4.6, we present the word-error rate of the Stockholm dialect trained system for several Scanian-dialect test speakers. We can see that the improvement in terms of performance when the combined method is used for 198 and 3814 adaptation sentences is almost uniform across the speakers, which verifies the assumption that there is a consistent mismatch across speakers of these two different dialects.

To compare the robustness and trainability of the standard ML training and adaptation algorithms, we performed training and adaptation experiments using fewer speakers in the training set, specifically 12 and 6 speakers. We use the term trainability above to refer to the ease with which a dialect-specific system can be developed. Clearly, the capability of developing a dialect-specific system with as few training speakers as possible is desirable, since it saves both time and money.

The smaller subsets of speakers were selected randomly out of the total number of 31 speakers available in the initial training set, and were equally divided across the two genders. We tried to select speakers from all 4 sub-dialects, so that the resulting system remains subdialect-independent. The results are illustrated in Figure 4.2. We see that for standard ML training, the error rate is very large when fewer than 1000 sentences from 31 speakers are used. Moreover, the ML training error rate is getting even larger as the number of speakers in the training set decreases. For example, if we use roughly 500 training sentences, the 31-speaker error rate increases by 9% and 29% when sentences from 12 and 6 speakers are considered, respectively. On the other hand,

Speaker	*Word Error Rate %*		
	Non adapted	Meth.II+Bayes 198 sent.	Meth.II+Bayes 3814 sent.
d09	24.94	8.53	8.31
d0b	27.05	12.32	9.90
d0k	21.92	8.49	5.42
d0j	28.64	9.24	6.70
d0r	29.85	13.93	6.71
d0v	19.72	7.66	5.10
d12	26.29	10.07	6.39
d13	22.88	5.26	2.75
total	25.08	9.37	6.40

Table 4.6: Word recognition performance across Scanian-dialect test speakers using non-adapted and combined-method adapted Stockholm dialect models

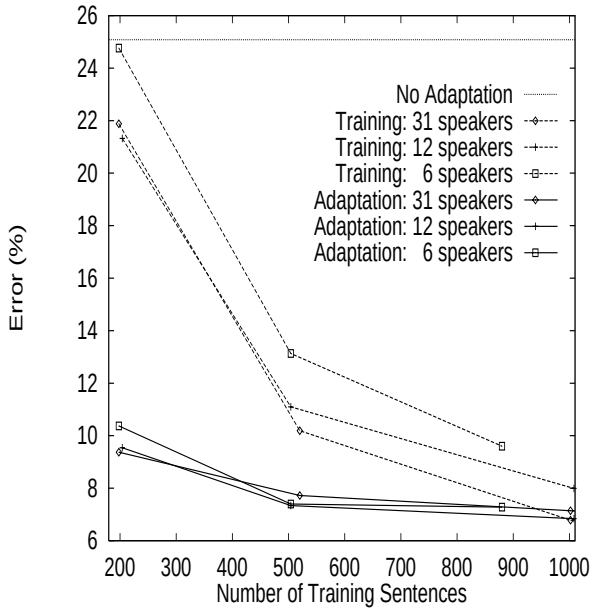


Figure 4.2: Comparison of dialect training and adaptation results for different number of speakers.

for the dialect-adapted system, the error rate using 12 and 6 speakers in the adaptation data remains as small as when using the full set of 31 speakers. The small differences are within the statistical error.

The reason for the significantly better performance of the adaptation schemes over standard ML training for small number of speakers is that speaker variability in the systems developed using adaptation techniques is captured from the prior knowledge, which the systems trained using standard ML techniques lack. In general, when we compare adaptation and training results we can conclude that adaptation significantly outperforms training for small amounts of sentences, and small number of speakers. For example, when we perform training with 31 speakers and 520 sentences the results obtained are similar with the adaptation experiments with as few as 6 speakers and only 200 training sentences. Similarly, the performance of a system trained with 31 speakers and 1000 sentences is similar to that of a system trained with only 6 speakers and 500 sentences. Therefore, both the robustness and trainability of an adaptation-based system are highly increased, when compared to standard ML training.

4.4.3 Summary

In this Section we have discussed the issue of dialect mismatch in an ASR system. We found, for the pairs of dialects that we examined, that there is a consistent degradation in performance across speakers when there is a dialect mismatch. Hence, we selected to improve the performance of the system using adaptation methods. We tested transformation and combined transformation and Bayesian adaptation algorithms to adapt a Stockholm-trained system to the Scania dialect. The results showed that adaptation is capable of improving the robustness of our system, and that the performance of the adapted system improved dramatically over the mismatched condition with very small amounts of adaptation data. Moreover, we showed that the recognition performance of the adapted system does not degrade when we reduce the number of different speakers from which the training data was collected, something not true for standard ML training. Hence, in terms of robustness and trainability, adaptation is a much better alternative for the development of dialect-specific systems than standard ML training.

4.5 Language Modeling

In language modeling for speech recognition systems, sparse training data has been one of the main problems. In this project, we investigated three approaches to tackle this problem, that is, interpolating in-domain data with out-of-domain data, using class-based language models (LMs) mixed with word bigram models, and splitting compounds for the Swedish System. For the last two approaches, we see some promising improvements.

4.5.1 Interpolating In-domain LMs with Out-of-domain LMs

Successful statistical language modeling requires large amounts of domain-specific data, which is not always available. One way to remedy this is to interpolate small

amounts of available domain-specific data with large amounts of domain-general data. The interpolation is the weighted average of the two bigram probabilities from the two corpora. We completed one quick experiment for the Swedish recognition system. Results show that a small improvement was obtained after interpolating with Swedish News texts. The WER changed from 8.88% to 8.70%, which is not a significant reduction.

4.5.2 Class-Based Language Modeling

We have implemented a class-based bigram LM (Brown *et al.*, 1992), and tested it with our English and Swedish ATIS systems. The English classes were defined manually, and they include city names, airlines, and airline codes. The Swedish classes were basically translated from English with some minor modifications. Results show that with large amounts of training data (20,000 sentences), pure class-based bigram LMs perform as well as word bigram LMs, while with small amounts of training data (5,000 to 10,000 sentences), pure class-based bigram LMs outperform word bigram LMs significantly, and the interpolated word bigram and class bigram systems gave even further improvement over pure class bigram systems (see Table 4.7). Results also show that the interpolated systems perform as well as the word bigram systems that are trained with twice the amount of data. Compared with existing recognizers that have word bigram LMs trained with large amounts of data, the new English system with class-based bigram LMs mixed with word bigram LMs reduced the WER from 7.02% to 6.37% (4660 words) (see Table 4.8), and the new Swedish system reduced the WER from 7.90% to 7.66% (totally 3758 words) (see Table 4.10). For low-WER systems, the improvement is quite impressive. We suspect that the relatively small improvement for the Swedish system is partially due to the limited variability of our testing set. A better test set for the Swedish system is under construction at Telia. For both Swedish and English, we have significant perplexity (PPL) reductions by using interpolated class-based LMs (see Tables 4.9 and 4.11).

We also experimented with a multiple-stage backoff scheme. This scheme uses class bigrams where word bigrams are absent in training, and it backs off further to the corresponding unigram word probabilities when class bigrams are absent in training. The PPL for this new scheme shows a moderate improvement over the class-based LMs, but its WER increases.

4.5.3 Compound Splitting in the Swedish System

In many languages, including German, Dutch, Swedish, Finnish and Greek, compound nouns can be formed by concatenation of single nominals. Because of the nature of the productivity in compounds, it is less favorable to simply list compounds in the lexicon. For example, in an experiment on SQALE training texts (Lamel *et al.*, 1995, page 186), use of 20,000-word lexicons for both German and English resulted in a 7.5 % out-of-vocabulary (OOV) rate for German and 2.5 % for English. The German lexicon had to be extended to 64,000 words to obtain OOV rates similar to those of the 20,000-word lexicon for English. We used a simple splitting method and compared it with its unsplit counterpart. For the 'split' version, all compound words, including numbers, were

Results on English ATIS			
	Training sentences	Test 1 (2000 words)	Test 2 (4660 words)
WordBi	23K	8.95%	7.02%
ClassBi	23K	8.85%	6.91%
WordBi	15K	9.25%	7.38%
ClassBi	15K	9.15%	7.10%
WordBi	10K	9.40%	7.73%
ClassBi	10K	9.25%	7.40%
WordBi	5K	10.25%	8.26%
ClassBi	5K	9.50%	7.40%
WordBi+ClassBi	10K	8.75%	7.10% (not tuned)
WordBi+ClassBi	5K	9.35%	7.21% (not tuned)

WordBi: Word bigram LM system
ClassBi: Class bigram LM system
WordBi+ClassBi: Word bigram interpolated with Class bigram LM system

Table 4.7: Word error rates of word bigrams vs. class bigrams with respect to different amounts of data.

Results on English ATIS			
	Test 1 (2000 words)	Test 2 (4660 words)	Ratio
Baseline	8.95%	7.02%	N/A
Class-gram	8.85%	6.91%	N/A
Word+class	8.15%	6.52%	0.1
Word+class	8.20%	6.37%	0.2
Word+class	8.25%	6.42%	0.3
Word+class	8.30%	6.52%	0.5
Word+class	8.50%	6.57%	0.7
Word+class	8.45%	6.65%	0.9
ClassBO	9.80%	8.37%	

Baseline: PTM system with PPR
Class-gram: Pure class bigram LM system
Word+class: interpolated word bigram and class bigram LM system
ClassBO: Multi-stage class backoff LM system

Table 4.8: Word error rates of word bigram model, class bigram model, and interpolated models for English

Perplexity (PPL) results on English ATIS		
	PPL	Ratio (OOV rate is 9/8884)
Word bigram	33.0	N/A
Class bigram	25.4	N/A
Word+class	24.6	0.4 (tuned, PPL range: 24.6-25.5)
ClassBO	23.3	0.1 (tuned, PPL range: 23.5-23.3)

Word+class: interpolated word bigram and class bigram LM system
ClassBO: Multi-stage class backoff LM system

Table 4.9: PPLs of word bigram model, class bigram model, and interpolated models for English.

Results on Swedish ATIS		
	444 sentences (3758 words)	Ratio
Baseline (split system)	7.90%	N/A
Class-gram	8.01%	N/A
Bi+Class	7.64%	0.3 (tuned on interpolating weights)

Table 4.10: Word error rates of word bigram model, class bigram model, and interpolated models for Swedish

Perplexity (PPL) results on Swedish ATIS		
	PPL	Ratio (OOV rate is 0/3758)
Baseline (split system)	40.260	N/A
Class-gram	22.208	
Bi+Class	20.715	0.3 (tuned on interpolating weights)

Table 4.11: PPLs of word bigram model, class bigram model, and interpolated models for Swedish

	Split	Unsplit
WER with respect to compound components (method 1)	7.9%	8.2%
WER with respect to full compounds (method 2)	8.3%	8.7%

Table 4.12: Word error rates of split vs. unsplit compounds for Swedish

manually split into their components, while for the 'unsplit' counterpart, only numbers were split. We conducted two experiments, and results show a promising improvement.

In one experiment, split training data and a split lexicon were used for language modeling; in the other, unsplit training data and an unsplit lexicon were used. The results are shown in Table 4.12.

Since the total number of words is different in the split and unsplit cases, the WER with respect to compounds can be measured in two different ways. In the first one, corresponding to the first row of Table 4.12, a one-to-one splitting function, which (for the purpose of the experiments) is used for mapping compounds to their components, was applied to both the hypotheses from the recognizer and the references. This function modifies only the unsplit data. We then compared the newly formed hypotheses and references to get the WER. Thus, in this case the WER was calculated with respect to the compound components.

In the second method, corresponding to the second row of the table, the same splitting function, but with mappings of numbers removed, was used in the reverse direction to map all the compound components in both hypotheses and references back to their compounds. The result was then used for computing the WER. Thus, in this case the WER was calculated with respect to the full compounds.

According to both measures, the WERs are lower for the split system. For some applications, such as dictation, what we care about is the direct output from the recognizer. In this case, the WER for the unsplit system is 9.3 %, compared to 7.9 % for the split system.

4.6 Development of a phone backtrace for the n-best decoding algorithm

The different variations of the various N-best algorithms already implemented in the Decipher speech recognition engine (sentence dependent and word dependent) only allow to get the string of words for each sentence hypothesis. For some applications (like segmental rescoring) it is necessary to have a decoded phone backtrace, i.e. the phone sequence and its segmentation, associated to each sentence hypothesis.

To obtain the phone segmentation for each hypothesis it is always possible to forced-align each sentence hypothesis to the actual speech utterance, and get the cor-

responding phone backtrace using the Viterbi algorithm state decoding. Nevertheless, for N-best lists in the order of hundreds or thousands of hypotheses, this is very time consuming. A more efficient solution is to augment the capabilities of our implementation of the word dependent N-best algorithm to include a phone backtrace for each sentence hypothesis. In this way both the N-best hypotheses and the corresponding phone segmentations are obtained in a single pass.

We achieve this goal by using a combination of the word-dependent and lattice N-best algorithms (Schwartz and Austin, 1991) along with storing extra information in the backtrace memories regarding the phone transitions within words.

The forward pass of the search is implemented according to the word-dependent n-best algorithm (Schwartz and Austin, 1991), while the backtrace procedure is an extended version of the one defined in the lattice n-best algorithm (Schwartz and Austin, 1991).

The word-dependent n-best algorithm is based in the fact that the best starting time for a word does depend on the previous word but most likely does not depend on the words before that. Thus, histories are distinguished based on only the previous word rather than the whole preceding sentence (as in the exact sentence dependent n-best algorithm) (Chow and Schwartz, 1990). At each state within a word we preserve the total probability (and pointers to the corresponding paths) for each of the n different preceding words. (Note that n is in general much smaller than N, N being the number of desired final hypotheses).

This method requires the grammar processor to propagate multiple hypotheses for each state in each word model, one for each predecessor word. This uses extra memory in the grammar processor for multiple active hypothesis structures. So the search space in the grammar processor is increased due to multiple active hypotheses for each node, according to the previous word.

At the end of a word we record in the backtrace memory the scores for the paths coming through each of the previous word hypotheses, also we record the name of the previous word. Then we proceed with a single history with the name of the word that just ended.

Also, when we exit the final state of each phone within a word, we store in the backtrace memory the names and the times of the ending phones. This information will allow us to recover the phone segmentations for each hypothesis.

At the end of the sentence we perform a recursive backtrace to derive the list of most likely sentences which is similar to the lattice n-best algorithm described in the literature (Schwartz and Austin, 1991).

The steps of the extended n-best with phone backtrace algorithm may be describes as follows:

1. Run the forward word dependent N-best. Each node may correspond to multiple hypotheses in the active array. Fill the backtrace memories with word ending times and probabilities. Add also phone ending times and corresponding probabilities.

2. Create the initial answer list by placing all words that ended the sentence onto the answer list. The backwards score for each of these partial answers is just the word ending score.

3. For each answer, try to extend the answer by one more word towards the beginning of the sentence. The extension words we try to connect to are all the words that ended at the start time of the current answer, and have the same node as the best predecessor word. The score for an extended answer is the original score, plus the difference between the extension word and the best extension word. In doing so we disallow connections whose forward plus backward scores fall below a pruning beamwidth of the best final sentence score. This pruning keeps the algorithm from blowing up by trying to add too many extensions.

4. While extending a path for a given word, backtrace also the corresponding phone sequence in terms of phone labels and ending times for the phones.

5. Keep extending all answers on the N-best list until no more extensions are possible, i.e., all answers span the entire sentence.

6. Sort the answers, remove any duplicates, and return up to the requested number of answers.

The experimental evaluation of the new extended algorithm showed almost the same speed than the previous word-dependent n-best search algorithm, the only significant change was the memory required to store the needed extra information in the backtrace, still it was within a reasonable size for the ATIS type task.

4.7 The Bilingual Speech Recognition System

We developed a multilingual speech recognizer capable of decoding a word string in any of a given set of languages. Language identification is achieved simultaneously, as a result of observing the language identity of the majority of the hypothesized words. Our approach is to treat all words as equal tokens regardless of the languages they belong to. The statistics of the acoustic and language models are estimated using a multilingual speech database with orthographic transcriptions. Language-specific knowledge is incorporated into the system through the dictionary of pronunciations used by the HMMs and by specifying phoneme classes that may contain phonemes for different languages. In this initial system, the phoneme sets do not overlap across languages. The proposed approach has some interesting characteristics:

- HMMs of allophones (of any language) that belong to the same classes and share similar contexts, could potentially share the same Gaussian codebooks. This work investigates the effect of Gaussian sharing on recognition performance.

- Multilingual language models can be used for improving language identification performance. It allows us to incorporate a high-level knowledge source for language identification at the lexical level.

- The system is capable of recognizing sentences spoken in more than one language. Mixing words from different languages, or "code-switching", is common within linguistic communities where there is general familiarity with more than one language.

- In real-time multilingual applications, a single decoder can be used. Alternative approaches usually do language identification followed by a language-specific recognizer or require multiple recognizers to run in parallel.

4.7.1 Experimental Setup

We experimented using a bilingual (English/Swedish) recognizer for the "Air Travel Information System" (ATIS) domain. For rapid experimentation we limited and balanced the amount of training data to 4000 male utterances per language. We organized our experiments based on the sharing of model parameters at the acoustic and language levels. For acoustic modeling, we used phonetically tied mixture (PTM) and Genonic models (Digalakis *et al.*, 1996). The PTM phoneme classes were organized based on place of articulation for vowels and manner of articulation for consonants. The English and Swedish phonemes were grouped according to the following classes: front vowels, central vowels, back vowels, dipthongs, semivowels and glides, nasals, voiced fricatives, unvoiced fricatives, affricatives, aspirated, voiced plosives, and unvoiced plosives. Swedish HMM allophones of a given class share the same Gaussian codebook. In a Genonic system, the sets of HMM states that share the same mixture components are determined automatically using agglomerative clustering techniques (Digalakis *et al.*, 1996). We trained mono- and multilingual systems. Monolingual systems share no parameters across languages while multilingual systems may share Gaussian codebooks. Statistical grammars were constructed in the form of backoff bigram language models (Katz, 1987). Monolingual language models were trained using text from a single language. Multilingual language models were trained using all the English and Swedish data pooled together. The latter resulted in a bilingual LM with a single backoff node. Using a single backoff node allows hypotheses to contain words in both languages. We are also planning to evaluate the case in which hypotheses are constrained to have all words in the same language.

4.7.2 Multilingual Recognition

The multilingual PTM system with shared acoustic parameters uses 12 phoneme classes. In this case, phonemes in the same classes share the same Gaussian codebook. The non-shared PTM system is trained using 24 classes. Each language-specific set of phonemes has a separate codebook. The Genone system is booted from the corresponding PTM system. The shared system has twice as many Gaussian components as the non-shared system, to maintain a constant ratio of Gaussian components to training vectors. The initial results are summarized in Table 4.13. In the table, shared acoustic model means that Gaussian codebooks are shared across languages. Shared language model means that the LM shares a backoff node and the search space covers both languages. The genonic system significantly outperforms the PTM system in most cases because of the greater number of parameters. The difference in accuracy between the PTM and genonic systems is limited by the total amount of training data available (4000 utterances per language). We also observe that sharing acoustic parameters does not seem to affect the word-error rate. Sharing a backoff node in the language model results in significant degradation in performance. This degradation is more significant for the

Test Language	Shared Acoustic Model	Shared Language Model	PTM	Genones
English	No	No	7.7	7.5
English	Yes	No	8.0	7.4
English	No	Yes	7.9	7.9
English	Yes	Yes	8.1	8.1
Swedish	No	No	7.5	6.5
Swedish	Yes	No	7.8	6.7
Swedish	No	Yes	8.4	7.6
Swedish	Yes	Yes	8.8	7.6

Table 4.13: English/Swedish word error rates for various speech recognition systems

Test Language	Number of Training Sentences	Vocab Size	OOV (%)	Perplexity	
				Non-Shared LM	Shared LM
English	20K	1165	0.2	22.4	23.8
Swedish	11K	1266	0.3	14.9	17.7

Table 4.14: Comparison of English and Swedish language models

Swedish test set: 6.5% to 7.6% for the genonic system with no acoustic parameter sharing. The same case in English results in an increase from 7.5% to 7.9%. This result could be associated with the greater increase in perplexity in the Swedish test set compared to the English test set (see Table 4.14). The increase in error rate could also be explained by the imbalance in the amount of data used for training the language models.

4.7.3 Language Identification

We also analyzed the language identification performance of the bilingual system. In Table 4.15, we show the percentage of words and sentences that contain a word in the other language. We observe that about 1% of the recognized words have the wrong language identity. Therefore, sharing the LM backoff node provides the flexibility of mixing languages in an utterance at the expense of an increase in error rate. Good language ID performance can easily be obtained by taking a simple majority on the words of a hypothesis (Table 4.16).

Test Language	Non-Shared Acoustic Models		Shared Acoustic Models	
	Word Miss (%)	Sent Miss (%)	Word Miss (%)	Sent Miss (%)
English	0.3	2.5	0.9	5.0
Swedish	1.1	6.4	0.9	4.9

Table 4.15: Language identification errors for words and sentences

Test Language	Non-Shared Acoustic Models Sent Miss (%)	Shared Acoustic Models Sent Miss (%)
English	0.0	0.2
Swedish	0.7	0.4

Table 4.16: Language identification errors after taking simple majority of words in hypothesis

4.7.4 Summary

We investigated the effect of sharing acoustic and language parameters for multilingual speech recognition. To improve multilingual performance, we are planning to constrain the language model to require that all the words in a hypothesis have the same language identity. We are also working on the optimization of the unconstrained bilingual language model, and on new approaches for sharing acoustic models across languages.

Chapter 5

Overview of Language Processing

David Carter, Manny Rayner and Mats Wirén

5.1 Introduction

When attempting to implement a language-processing architecture for any kind of practically useful speech understanding system, there is a tension between two fundamental requirements. Other things being equal, we would like our language processing to be based on declarative, linguistically motivated descriptions of language; this gives us the advantages of increased portability across domains and (to a lesser extent) languages, and makes it easier to incorporate insights from theoretical linguistics into the system. However, important as these goals are, it is even more important that the system be at least moderately fast and robust. A system which is too slow and brittle is not of great practical interest, even if it has a theoretically impeccable pedigree.

There is at the moment wide-spread disenchantment with the idea of building systems based on large hand-coded grammars. Critics of the approach generally offer some variant of an argument which can briefly be summarized as follows:

1. Grammars take too long to develop.
2. They always leak badly.
3. They need substantial manual tuning to give reasonable coverage in a new domain.
4. Even after doing that, processing is still very slow.
5. At the end of the day, performance is anyway no better than what you would get from a surface processing method.
6. So why bother?

We do not think these objections are unreasonable; they are based on many people's painful experience, including our own. However, we do not believe either that the problems listed above are insurmountable. This section gives an overview of the methodology we have developed for attacking them. In equally brief form, our response is:

1. We keep our grammars, lexica and other linguistic descriptions as general as possible, so that the large development cost is a one-off investment.

2. We make sure that the grammar contains all, or nearly all, of the difficult core constructions of the language; then most coverage holes are domain-specific, and relatively easy to locate and fix. See Chapters 9, 10 and 11.

3. Non-trivial tuning is needed when adapting the grammar for use in a specific domain. However, a large portion of this tuning can be performed semi-automatically with supervised training procedures usable by non-expert personnel (see Chapter 7). The remaining work can be organized efficiently using balanced corpora to direct expert attention where it will be most productive; see Section 8.

4. Automatic corpus-based tuning of the language description by grammar specialization and pruning makes grammar-based language processing acceptably efficient; see Chapter 6.

5. Bottom-up processing strategies can intelligently combine the results of "deep" linguistic processing and fall-back processing using shallow surface methods; see Section 5.2.

6. The results show a clear improvement over those produced by the surface methods alone.

In the remainder of this chapter, we will focus on the issues of portability, speed and robustness in the context of the Spoken Language Translator's hybrid language-processing architecture which was described in Section 1.2. Section 5.2 discusses those aspects of the system concerned with robust parsing with domain-specialized linguistic descriptions. Section 5.3 describes the semi-automatic domain adaptation process, and Section 5.4 concludes.

5.2 Linguistically Motivated Robust Parsing

Section 1.2 described how grammar-based parsing contributes to the general robust translation scheme. We now consider in more detail the question of how a corpus can be used to specialize a general grammar so that it delivers practically useful performance in a given domain. Can such a grammar be concretely useful if we want to process input from a *specific* domain? In particular, how can a parser that uses a general grammar achieve a level of efficiency that is practically acceptable?

The central problem is easy to state. By the very nature of its construction, a general grammar allows a great many theoretically valid analyses of almost any non-trivial sentence. However, in the context of a specific domain, most of these will be extremely implausible, and can in practice be ignored.

One possible solution is of course to dispense with the idea of using a general grammar, and simply code a new grammar for each domain. Many people do this, but one cannot help feeling that something is being missed; intuitively, there are many domain-independent grammatical constraints, which one would prefer only to need to code once. In the last ten years, there have been a number of attempts to find ways to automatically adapt a general grammar and/or parser to the sub-language defined by a suitable training corpus. For example, Briscoe and Carroll (1993) train an LR parser based on a general grammar to be able to distinguish between likely and unlikely sequences of parsing actions; Andry *et al.* (1994) automatically infer sortal constraints, that can be used to rule out otherwise grammatical constituents; and Grisham *et al.* (1984) describes methods that reduce the size of a general grammar to include only rules actually useful for parsing the training corpus.

Our work on parsing (see Chapter 6 for a fuller description) is a logical continuation of two specific strands of research aimed in this same general direction of focusing the search on only a small portion of the space of theoretically valid grammatical analyses.

The first is the popular idea of *statistical tagging* e.g. DeRose (1988), Cutting *et al.* (1992) and Church (1988). Here, the basic idea is that a given small segment S of the input string may have several possible analyses; in particular, if S is a single word, it may potentially be any one of several parts of speech. However, if a substantial training corpus is available to provide reasonable estimates of the relevant parameters, the immediate context surrounding S will usually make most of the locally possible analyses of S extremely implausible. In the specific case of part-of-speech tagging, it is well-known (DeMarcken, 1990) that a large proportion of the incorrect tags can be eliminated "safely", i.e. with very low risk of eliminating correct tags. We generalize the statistical tagging idea to a method called "constituent pruning"; this acts on local analyses (constituents) for phrases normally larger than single-word units. Constituents are pruned out if, on the basis of supervised training data (see Section 5.3.2 below), they seem unlikely to contribute to subsequent parsing operations leading to an optimal analysis of the full sentence. Pruning decisions are based both on characteristics of the constituents themselves and on the tags of neighbouring constituents. From each constituent and pair of neighbouring constituents, a *discriminant* (abbreviated description of the constituent or pair) is extracted, and the number of times this constituent or pair has led to a successful parse in training is compared to the number of times it was created (Dagan and Itai, 1994; Yarowsky, 1994). Constituents that are never or very seldom successful on their own, or that only participate in similarly unpromising pairs, are pruned out, unless this would destroy the connectivity of the chart.

The second idea we use is that of *Explanation-Based Learning* (EBL; Mitchell *et al.*, 1986; van Harmelen and Bundy, 1988). We extend and generalize the line of work described in Rayner (1988), Rayner and Samuelsson (1990), Samuelsson and Rayner (1991), Rayner and Samuelsson (1994) and Samuelsson (1994b). Here, the basic idea is that grammar rules tend in any specific domain to combine much more frequently in some ways than in others. Given a sufficiently large corpus parsed by the original, general, grammar, it is possible to identify the common combinations of grammar rules and "chunk" them into "macro-rules". The result is a "specialized" grammar; this has a

larger number of rules, but a simpler structure, allowing it in practice to be parsed very much more quickly using an LR-based method (Samuelsson, 1994a). The coverage of the specialized grammar is a strict subset of that of the original grammar; thus any analysis produced by the specialized grammar is guaranteed to be valid in the original one as well. The practical utility of the specialized grammar is largely determined by the loss of coverage incurred by the specialization process. We show in Rayner and Carter (1996) that suitable "chunking" criteria and a training corpus of a few thousand utterances in practice reduce the coverage loss to a level which does not affect the performance of the system to a significant degree (Rayner and Carter, 1996).

The two methods, constituent pruning and grammar specialization, are combined as follows. The rules in the original, general, grammar are divided into two sets, called *phrasal* and *non-phrasal* respectively. Phrasal rules, the majority of which define fairly simple noun phrase constructions, are used as they are; non-phrasal rules are combined using EBL into chunks, forming a specialized grammar which is then compiled further into a set of LR-tables. Each chunk applies at a particular level of parsing, depending on the kind of constituent it can create. Parsing proceeds bottom-up by interleaving constituent creation and deletion. First, the lexicon and morphology rules are used to hypothesize word analyses. Constituent pruning then removes all sufficiently unlikely edges. Next, the phrasal rules are applied bottom-up, to find all possible phrasal edges, after which unlikely edges are again pruned. Finally, the specialized grammar is used to search for constituents at successively higher levels; pruning may be carried out after any of these levels has been completed, the decision depending on whether pruning at a given level offers an overall speedup for the domain in question.

5.3 Semi-automatic domain adaptation of grammars

Section 5.2 described how a general grammar can be made to deliver useful performance, measured in terms of speed and robustness, within a given domain. We now discuss the related question of how to achieve acceptable coverage. Our experience is that when an unmodified general grammar is used to process utterances from a given domain, it fails badly in two respects. Firstly, there are virtually always a number of serious coverage holes, reflecting constructions common in the domain which are inadequately handled by the grammar. Secondly, there is the ubiquitous problem of ambiguity; even when the coverage holes are fixed, most utterances receive multiple analyses, of which only a small proportion are correct.

In the remainder of the section, we discuss these two problems. In Section 5.3.1, we describe a simple methodology which allows us rapidly to identify and fix the important coverage holes in the grammatical rule sets. Section 5.3.2 describes a supervised training method which attacks the problem of ambiguity.

5.3.1 Rational Development of Rule Sets

An unmodified general grammar generally delivers poor coverage in a specific domain. However, our experience is that most of the utterances which fail to parse do so because of a relatively small number of isolated problems; these are typically missing lexical

entries, missing grammar rules for idiosyncratic types of phrase common in the domain, and minor faults in existing rules. Grammar bugs of this kind are easy to fix. The real problem is identifying them quickly and efficiently, so that effort is focussed on bugs which significantly affect coverage in the given domain.

Our methodology is based on the idea of constructing a "representative subcorpus". By this, we mean a small subset of the main corpus, intelligently selected so as to exemplify the important domain constructions in descending frequency order. Our recipe for constructing representative subcorpora is roughly as follows, and can be used by non-experts who have some basic familiarity with linguistics.

1. Since the grammar will operate bottom-up, it is unnecessary to be able to analyze all utterances as complete units. Thus start by dividing long utterances into smaller pieces, which can reasonably be thought of as units to be translated separately.

2. Assign part-of-speech tags to the "split" utterances using some kind of tagger.

3. Group utterances into equivalence classes under the relationship of having the same tag-sequence.

4. Manually regroup the classes produced by the previous step where necessary. In some cases, this involves reclassifying utterances which were incorrectly tagged; in others, a group may be split into two or three smaller groups, if the relevant utterances are intuitively dissimilar enough. This step can be performed by non-experts at the rate of several thousand sentences a day, using a simple interactive tool.

5. For each of the new classes, manually designate an element which intuitively is "most typical" of the class. This step can also be performed quickly by non-experts using the same interactive tool.

6. Construct the "representative subcorpus" by selecting the designated element from each class. Order the results by the size of the classes represented.

Our experience is that by starting at the top of the representative subcorpus and working downwards, it is possible to fix the important coverage problems in a new corpus with an investment of only a few weeks of expert effort. The representative subcorpus is also a valuable resource for performing subsequent routine system testing.

5.3.2 Training by Interactive Disambiguation

We have already indicated how discriminants are used at run-time to decide which constituents should be pruned. Similar discriminants are also used to choose between alternative analyses for a sentence. However, deriving discriminant statistics involves selecting the correct analysis. This requires human intervention, and we would prefer the human in question not to have to be a system expert; but even for an expert, inspecting all the analyses for every sentence would be a tedious and time-consuming task. There may be dozens of quite detailed analyses that are variations on a small

number of largely independent themes: choices of word sense, modifier attachment, conjunction scope and so on.

It turns out that some kinds of discriminant can be presented to non-expert users in a form they can easily understand. For training on an utterance to be effective, we need to provide enough such "user-friendly" discriminants to allow the user to select the correct analyses, and as many as possible "system-friendly" discriminants that, over the corpus as a whole, distinguish reliably between correct and incorrect analyses and can be used for this purpose at run time, either in constituent pruning or in preferring one analysis from a competing set. Ideally, a discriminant will be both user-friendly and system-friendly, but this is not essential.

We have developed an interactive program, the TreeBanker (described in full in Section 7.3), which maintains a database of the discriminants that apply to the different analyses of each sentence in a corpus. It presents discriminants to the user in a convenient graphical form. Among the most useful discriminants are the major categories for possible constituents of a parse; thus for the sentence "Show me the flights to Boston" the string "the flights to Boston" as a noun phrase discriminates between the correct reading (with "to Boston" attaching to "flights") and the incorrect one (with it attaching to "show"). Other discriminants describe semantic triples of head, modifier and dependent (for example, "flight+to+Boston", which is correct, and "show+to+Boston", which is incorrect), and other information about analyses such as the sentence type or mood.

The user may click on any discriminant to select it as correct or incorrect. Typically there will be far more discriminants presented than the number of distinct differences between the analyses. The effect of this is that the user can give attention to whatever discriminants he finds it easiest to judge; other, harder ones will typically be resolved automatically by the TreeBanker as it reasons about what combinations of discriminants apply to which analyses. For example, when the CLE analyses the sentence "What is the earliest flight that has no stops from Washington to San Francisco on Friday?", it yields 154 analyses and 318 discriminants, yet the correct analysis may be obtained with only two selections. Selecting "the earliest flight ... on Friday" as a noun phrase eliminates all but twenty of the analyses produced, and approving "that has no stops" as a relative clause eliminates eighteen of these, leaving analyses which are both correct for the purposes of translation. 152 incorrect analyses may thus be dismissed in less than fifteen seconds.

5.4 Summary

This chapter has presented an overview of a methodology which, in our opinion, demonstrates that hybrid approaches based on general hand-coded grammars can be practically useful in the context of a realistic speech-understanding task like medium-vocabulary spoken language translation. In particular, we have addressed what we see as the key questions: achieving adequate speed, robustness and coverage within a specific domain, and adapting the general grammar to the domain without excessive effort.

In the following chapters we will return to all of these points in greater detail,

and present detailed performance figures to substantiate the claim that the methods described can indeed be justified in practical terms.

Chapter 6

Customization of Linguistic Knowledge

David Carter and Manny Rayner

In this chapter, we show how a general grammar may be automatically adapted for fast parsing of utterances from a specific domain by means of the two methods already outlined in Section 5.2: constituent pruning, and grammar specialization based on explanation-based learning. These methods together give an order of magnitude increase in speed, and the coverage loss entailed by grammar specialization is reduced to approximately half that reported in previous work. Experiments described here suggest that the loss of coverage has been reduced to the point where it no longer causes significant performance degradation in the context of a real application. We also discuss another task that requires customization: that of choosing the correct analysis from among the many possibilities resulting from parsing.

The chapter is structured as follows. First, we give an overview of the CLE's language analysis process, particularly the way it interfaces to speech recognition. Section 6.2 then describes the constituent pruning method. Section 6.3 describes the grammar specialization method, focusing on how the current work extends and improves on previous results. Section 6.4 explains how customized data is used in analysis choice. Section 6.5 describes experiments where the constituent pruning/grammar specialization method was used on sets of previously unseen speech data. Section 6.6 concludes and sketches further directions for research, which we are in the process of investigating.

6.1 Linguistic Analysis in the Core Language Engine

The analysis component of the CLE takes as input from Decipher an N-best list (typically N=5; see Section 16.1) of utterance hypotheses, each with an associated acoustic score. The CLE converts the list to a scored lattice of words, which it analyses morphologically, creating a well-formed substring table of lexical constituents. This table

Cost	Sentence hypothesis
0	list all flights leaving denver between eight p m and nine p m
176	list all flights leaving in denver between eight p m and nine p m
268	list all flights leaving denver between eight a p m and nine p m
368	list all flights leaving denver between eight p m and nine a p m
395	list all flights leaving denver and between eight p m and nine p m

0	list	all	flights	leaving	denver		between		eight	p		...
176					in	denver						...
268										a	p	...
368												...
395							and	between				...

Figure 6.1: N-best list and part of word lattice for example sentence

is parsed in several stages. At each stage, a different set of rules is applied to create further constituents. Before some of the stages (the choice depending on the effect on overall system efficiency), the table is pruned of constituents that, on the evidence of training, seem unlikely to contribute to a complete, correct interpretation. After the last stage of parsing, such interpretations are, when possible, extracted from the table. If there is more than one interpretation, one of them is selected, also on the basis of judgments made during training, for transfer to the target language and thence generation and synthesis. If no full interpretations are created (and, indeed, before any attempt is made to create them), partial analyses are also sent for transfer and generation, as explained in Section 1.2; however, for the sake of simplicity we concentrate here on the creation of full analyses.

For example, if the user says “List all flights leaving Denver between eight p m and nine p m”, the N-best list may be as shown in the top half of Figure 6.1. Each sentence hypothesis is shown with its associated acoustic cost (shortfall from the score of the top hypothesis). The figure also shows part of the word lattice to which the N-best list is converted.

After a lexical edge is created for every word analysis, the first stage of pruning is carried out. In this particular case, the data collated from training leads all the edges with non-zero acoustic costs to be pruned out. Some zero-cost edges with inappropriate syntactic categories are also pruned, e.g. “list” as a noun (rather than a verb) and “p m” as a sequence of letters (rather than a time marker).

The first stage of parsing then applies relatively low-level rules, such as those that create simple NPs from determiners and head nouns (“the flights”), and time phrases from numbers and time suffixes (“eight p m”, “nine p m”). In the next stage of pruning, the edges for component words of these particular phrases are deemed unlikely to make any further *direct* contribution to correct analyses – i.e. to serve as the immediate daughters in correct rule applications that have yet to be made – so they are pruned, along with a number of other edges, both lexical and parser-created.

After further parsing stages, between three and five complete analyses are produced

(the exact number depending on how the system is configured). One of these is selected as correct by applying preference data derived from training to successively eliminate its competitors; this process is described in more detail in section 6.4 below.

The analyses output by the CLE are quasi-logical forms (QLF; Alshawi (ed), 1992, Alshawi and Crouch, 1992). QLF is a version of first-order predicate logic augmented with constructs for unresolved referential, function word-sense, and quantifier-scope ambiguities. Differences between the QLFs for a sentence reflect different choices of content word senses and syntactic constructs. QLFs are constructed by semantic rules which are in a many-to-one correspondence with the syntax rules used in parsing; thus syntactic decisions such as PP attachments are reflected quite directly in QLFs.

6.2 Constituent Pruning

Before each of the CLE's parsing stages, the constituent table (henceforth, the chart) may be examined to locate edges that are relatively unlikely to make any further contribution to correct analyses – i.e., edges that are very likely either to be wrong, or only to be correct as daughters of other existing edges, and not to be directly useful as input to further parsing stages. When such edges are located, we remove them, or at some stages of parsing (because the creation of QLFs once parsing is complete relies on their continued existence) retain them but hide them from further parsing stages.

Currently, we prune before the phrasal and full parsing stages. When the stratified EBL grammar [REF] is used, pruning may take place between its stages too, although in the current implementation we do not do this because the resulting savings in parsing time are small enough to be outweighed by the time taken to prune.

For example, after the string "Show flight D L three one two" is lexically analysed, edges for "D" and "L" as individual characters are pruned because another edge, derived from a lexical entry for "D L" as an airline code, is deemed far more plausible. Similarly, edges for "one" as a determiner and as a noun are pruned because, when flanked by two other numbers, "one" is far more likely to function as a number.

Phrasal parsing then creates a number of new edges, including one for "flight D L three one two" as a noun phrase. This edge is deemed far more likely to serve as the basis for a correct full parse than any of the edges spanning substrings of this phrase; those edges, too, are therefore pruned. As a result, full parsing is very quick, and only one analysis (the correct one) is produced for the sentence. In the absence of pruning, processing takes over five times as long and produces 37 analyses in total.

6.2.1 Discriminants for Pruning

Our algorithm estimates the probability of correctness of each edge: that is, the probability that the edge will contribute directly (as an immediate daughter of a correct edge to be built later in parsing) to a correct full analysis of the sentence, given certain lexical and/or syntactic information about it. Each piece of information used as a criterion for a pruning decision can be viewed as a *discriminant* (Dagan and Itai, 1994; Yarowsky, 1994) because it potentially discriminates between correct and incorrect analyses. As

Section 6.4 will explain, discriminants (of a different kind) are also the basis of the preference component of the CLE which chooses between competing QLF analyses.

In both the pruning and preference phases, we would like to estimate the probability that when a discriminant occurs, it characterizes (what will eventually become) the correct analysis of the input. As a shorthand, we call such occurrences *good* ones, and estimate their probabilities using counts of good and bad occurrences encountered during training (see Chapter 7). For a pruning discriminant d, a good occurrence is when d is true of (i.e. describes a part of) the syntactic parse tree eventually used to create the correct analysis of the input, and a bad occurrence is when d occurs in the chart but does not contribute to the correct analysis. If d is a preference discriminant, a good occurrence is when it is true of the correct analysis but not of every analysis, and a bad occurrence is when it is true of some analyses but not of the correct one. In both cases, the probability estimate is a smoothed version of the maximum likelihood estimate,[1] i.e. of the number of good occurrences divided by the total number of occurrences, both good and bad.

We describe in Section 6.2.3 below the way in which the probability estimates for different discriminants are calculated, taking into account data sparseness and the influence of acoustic scores. For the moment, we will take these estimates as given.

The current discriminants used in pruning, each associated with an edge in the chart, are as follows:

- its *right bigram(s)*. A right bigram is a sequence consisting of the current edge followed by an edge immediately to its right, considering only the following data about each of the two edges:
  - its *class*. For a lexical edge, the class is the *semantic word class* pre-defined for the word in question: words with similar distributions, such as city names, are grouped into classes to overcome data sparseness. If no explicit class is defined for the word, the word itself is used. For for a non-lexical edge, the name of the final (topmost) *grammar rule* that was used to create it is used as the class.
  - its *tag*: usually its major category symbol, although a few categories correspond to several tags to allow additional distinctions derived from feature values to be represented.
- the *left bigram(s)* for the edge: as above, but considering the current edge and one of its left neighbours.
- The "*tree-gram*" for the tree of grammar rules, with words at the leaves, that gave rise to the edge in question. The tree-gram for a tree is a tuple consisting of the class and tag (see above) of the leftmost terminal node, the mother (root) node of the tree, and the rightmost terminal node. For example, the tree-gram for a noun-phrase analysis of the phrase "the first Delta flight" would be

```
(the/detn, np_det_nbar/np, flight/nbar)
```

[1] The maximum likelihood value of a parameter is that value which maximizes the likelihood of observing the data that have in fact been observed. This is *not* the same thing as the most likely value of the parameter.

Here, the leftmost and rightmost daughters "the" and "flight" are their own classes; the tags are the major category symbols, except that the tag `detn` stands for a `det` (determiner) which cannot function on its own as a noun phrase (contrast "which").

Other discriminants, such as trigrams, finer-grained tags and alternative definitions of tree-grams, are obviously possible, and could be applied straightforwardly within the framework described here. The system in fact allows trigrams to be applied, but on ATIS data, their use seems unnecessary. When a test was run on 100 unseen test English N-best lists, and trigram-based pruning was applied after tree-gram and bigram-based, only 5% more constituents were pruned out. However, there were nearly three times as many trigram-based pruning rules (i.e. trigram discriminants with sufficiently low probability estimates) as tree-gram and bigram ones put together, and using trigrams more than doubled the time required for the first pruning stage.

The score (probability estimate) for an edge is defined as the minimum of its tree-gram score, its maximum left-bigram score over all the possible choices of left neighbour, and its maximum right-bigram score over all the possible choices of right neighbour. This is justified as follows.

Firstly, we take the maximum over possible neighbours because a correct edge participates in only one left bigram and one right bigram (at a given level) in the winning parse tree. Thus, for example, "to" can function either as preposition ("`p`") or as an infinitive marker (encoded as a verb, "`v`", in the CLE grammar of English). In the sentence "Show me flights to that airport", the bigram `(to/v,that/detn)` scores very badly, but this should not lead to the determiner edge for "that" being penalized, because the bigram `(to/p,that/detn)` has a much better score, and it is in fact this bigram that characterizes the correct parse. If we did penalize the determiner edge for "that" because of the first bigram, we would run the risk of reducing its scores to near those of the other (noun phrase and complementizer) edges for "that" and therefore failing to prune those two edges.

Secondly, taking the overall minimum involves making an assumption of maximal statistical dependence (Yarowsky, 1994), rather than the more common assumption of full independence. If events $E_1, E_2, ..., E_n$ are fully independent, then the joint probability $P(E_1 \wedge ... \wedge E_n)$ is the product of $P(E_1)...P(E_n)$, but if they are maximally dependent, it is the minimum of these values. Of course, in this and most other situations, neither assumption is any more than an approximation to the truth. But assuming dependence makes sense because there is likely to be a fair amount of dependence between discriminants that all reflect local syntactic information. Also, on the basis of local information alone, such as that reflected in the pruning discriminants, it is not usually possibly to predict with confidence that a particular edge is highly *likely* to contribute to the correct analysis (since global factors will also be important) but it often is possible to spot highly *unlikely* edges using one (or more) of the discriminants available. Extracting a discriminant from an edge involves taking a particular view of that edge by abstracting out a small (but, we hope, relevant) part of the information about it that is present in the chart. The resulting probability estimate reflects the number of times that edges which, on the view represented by the discriminant, are identical to the current one, have been judged right or wrong in training. If we can find a view of

(i.e. a discriminant for) an edge that identifies it with a set of training edges that are virtually always wrong, then we are justified in pruning it, regardless of the views of the edge represented by other discriminants. Also, taking minima means there is no need to include in the run-time system any discriminant scores that are not sufficiently low to trigger pruning; this much reduces the amount of data required at run-time.

A further advantage of taking minima is that if we do so, the estimate of the joint probability depends much less strongly on the number of events compared, and so estimates for alternative joint events can be directly compared, without any possibly tricky normalization, even if they are composed of different numbers of atomic events. This property is desirable: it means that further discriminants could be introduced that are only defined for some kinds of edge.

6.2.2 Deciding which Edges to Prune

At each pruning stage (before phrasal parsing, before full parsing, etc), discriminants, with their accompanying probability estimates, are calculated for each edge in the chart. An estimate of 1/200 or lower is taken as sufficient grounds for pruning an edge. Pruning is carried out on edges with successively higher estimates until either this threshold is reached, or (more rarely) pruning the next edge out would destroy the connectivity of the chart, i.e. would remove all the remaining complete paths through it and therewith the hope of a full parse.

For efficiency reasons, the implemented system in fact involves a small change to the algorithm so far described. We first calculate tree-gram scores, then prune out any edges judged sufficiently unlikely on tree-gram grounds alone (bearing in mind that because the overall probability estimate is defined as the minimum of the tree-gram and bigram scores, if the tree-gram estimate is below the threshold then the overall estimate will also be). For non-initial pruning stages, this often results in quite a lot of edges being pruned, which greatly reduces the number of bigrams to be examined, speeding up pruning considerably.

Left and right bigram scores are then calculated together, and those *bigrams* whose scores fall below the threshold are marked with their scores. All *edges* whose left bigrams are all marked, and/or whose right bigrams are all marked, are also then marked, as are any still-unmarked bigrams in which they participate. Scores are propagated in this way until no further change occurs, and then marked edges are pruned in order of increasing score.

6.2.3 Probability Estimates for Pruning

As already stated, the probability estimate for an edge (or N-gram of edges) according to a certain criterion (discriminant) is based on the numbers of occasions in training that edges with the same description according to that discriminant did and did not, at the parsing level in question, contribute directly to correct parses. Suppose, for example, that a given bigram discriminant was correct ("good") on G occasions and incorrect ("bad") on B. Then we would want an estimate for that discriminant of around $G/(G+B)$.

However, this fairly simple picture is complicated in three ways. Firstly, as always with tasks of this kind, smoothing is needed for low values of G and/or B. Secondly, and relatedly, for pruning to be as effective as possible we need to look at generalized discriminants: the specific bigram $(C_1/T_1, C_2/T_2)$ may have $G = 0$ and $B = 2$, which on its own hardly inspires enough confidence to justify a pruning decision, but it may be that the more general set of bigrams satisfying $(C_1/T_1, */T_2)$, where $*$ is any value of C_2, has $G = 0$ and $B = 300$, which presents a rather different picture. Thirdly, account must somehow be taken of the acoustic scores for different word candidates contributed by the recognizer – and, by implication, for edges constructed from them during parsing. We discuss each of these complications in turn.

Complication 1: Smoothing

Given G good (correct) occurrences of a datum such as a tree-gram or bigram, and B bad ones, the maximum likelihood estimate of a subsequent occurrence being good is $G/(G + B)$. However, such an estimate is not a good basis for pruning; in particular, if $G = 0$, it evaluates to zero for any positive B, however small or large. To smooth the estimate, we need to assume some prior underlying distribution for the events ("prior" in the sense of it being our best guess prior to seeing any occurrences), and take as our estimate the expected value of the posterior distribution given that prior and our observations of G and B.

Suppose we assume that the probability p_d that a given occurrence of a datum d will be good is itself distributed uniformly at random as d varies. That is, if we take a particular datum for which we have not yet seen any occurrences at all, then any probability between 0 and 1 that a given occurrence of d will be good is equally likely (but all occurrences of the same d are governed by the *same* value p_d). In other words, d is as likely to be a reliably good discriminant (p_d=0.99, say) as to be a fairly uninformative one (e.g. p_d=0.48) or a reliably bad one (say p_d=0.02). This (we assume) is the prior distribution for p_d: its density function is $f_d(p) = 1$ for all p from 0 to 1. The expected value of p_d (in other words, our estimate of the probability that a new occurrence of d will be good) is $\frac{1}{2}$.

Now suppose we observe G good occurrences of d and B bad ones. The density function $g_d(p)$ for the posterior distribution for p_d is given by the following standard formula, which is related to Bayes' rule:

$$g_d(p) = \frac{f_d(p)P(G, B|p)}{\int_0^1 f_d(x)P(G, B|x)dx} \tag{6.1}$$

The expected value of $g_d(p)$ is given by the formula

$$E(g_d) = \int_0^1 x g_d(x)dx \tag{6.2}$$

$$= \frac{\int_0^1 x^{G+1}(1-x)^B dx}{\int_0^1 x^G(1-x)^B dx} \tag{6.3}$$

since $P(G, B|x)$, the probability of getting G good outcomes and B bad ones from a binary distribution with probability x of a good outcome, is $x^G(1-x)^B$ times a

combinatorial constant which can be ignored here as it is the same for numerator and denominator of equation 6.1.

A standard result is that

$$\int_0^1 x^M y^N \, dx = \frac{M!N!}{(M+N+1)!}. \tag{6.4}$$

The formula for $E(g_d)$ therefore reduces to

$$E(g_d) = \frac{(G+1)!B!/(G+B+2)!}{G!B!/(G+B+1)!} \tag{6.5}$$

$$= \frac{G+1}{G+B+2} \tag{6.6}$$

which is our posterior estimate for the probability that a new occurrence of d will be good, given a uniform prior, G good observations and B bad ones.

It can be seen that for non-negative G and B, equation 6.6 gives $0 < E(g_d) < 1$, and $E(g_d)$ strictly increasing in G and strictly decreasing in B. These are desirable properties: no matter how many bad occurrences we observe, the possibility of the next one being good is never quite ruled out, but it is viewed as increasingly unlikely. In terms of pruning, this means that a datum with no good occurrences and 100 bad ones scores worse than one with no good occurrences and 10 bad ones, and is therefore viewed as a better candidate for pruning.

However, our initial conservative assumption of a uniform prior distribution turns out not to fit our population of data very well. Examination of training data suggests that there are relatively few data d such that p_d is near the middle of the range; values close to 0 and 1 (especially to 0) are much more common. The effect of this can be seen by comparing two hypothetical data, say d_1 with $(G, B) = (0, 8)$ and d_2 with $(G, B) = (9, 89)$. These both give $E(g_d) = 0.1$ according to equation 6.6, but in practice d_2 would be rather more likely to yield a good outcome on the next observation (after all, it has already done so nine times) than d_1, whose true underlying probability is likely to be closer to zero.

Equation 6.6 can be interpreted as saying that to estimate the probability that the next occurrence of d will be good, we should use the maximum likelihood estimate not for the events that we have actually observed (G good and B bad) but for a set of $G' = G + 1$ good and $B' = B + 1$ bad: i.e. the set we observed plus one more good event and one more bad (the symmetry reflecting the uniform prior). If we apply this strategy but take as our prior distribution what until now has been our posterior one, we get

$$G' = G + \lambda \frac{G+1}{G+B+2}$$

$$B' = B + \lambda \frac{B+1}{G+B+2}$$

where λ is the number of extra events assumed. Setting λ arbitrarily to one and substituting into the maximum likelihood formula $\frac{G'}{G'+B'}$ we get an estimate of

$$\frac{G(G+B+3)+1}{(G+B)(G+B+3)+2} \tag{6.7}$$

for the expected value of the (new) posterior distribution, i.e. for the probability that the next observed occurrence of d will be good. This formula has the desirable properties noted above (never zero, strictly increasing in G and decreasing in B) but seems to do better with cases like d_1 and d_2 above: if d_1 has $(G, B) = (0, 8)$ then a d_2 with $(G, B) = (1, 89)$, rather than $(9, 89)$, will be given a similar estimate. This looks intuitively better: training data with counts like these tend in practice to be about equally likely to yield a good outcome on the next trial. Furthermore, to fall below the threshold of $\frac{1}{200}$ required for pruning to occur, if $G = 0$ we need $B \geq 13$; for non-zero G, $B \geq 200 * G$ is approximately sufficient, and these values also seem to work well in practice.

Of course, the justification for formula 6.7 is only informal and it is possible that other formulae might give better results. However, our past experience with this kind of problem suggests that what is important is to find a formula which *qualitatively* reflects the nature of the data being processed; once this has been achieved, quantitative optimizations tend to be relatively unproductive. In fact, we have observed no failures in pruning (or preference calculations, where this formula is also used – see Section 6.4.1) caused by the formula yielding intuitively wrong values given the (G, B) values it is provided with. This was not the case for the earlier formula 6.6 based on the uniform prior.

Complication 2: Generalized Discriminants

Suppose the pruner comes across an edge for "you" as an NP followed by one for "M" as a character. In the ATIS domain, this seems fairly unlikely to be good (and may well result from a recognizer error). This particular bigram is in fact always bad when it occurs in the ATIS training data used for English; the problem is that it only occurs there twice, and *a priori*, counts of $G = 0, B = 2$ (giving a probability estimate of $\frac{1}{12}$ from equation 6.7) do not justify pruning. However, bigrams in which "you" as an NP is followed by *any* character occur a total of 22 times in training, with all the occurrences being bad ones; and bigrams for any NP followed by "M" as a character occur 471 times, again all bad. Either of these counts does justify pruning.

On the other hand, there are also cases where a maximally specific bigram (with all four fields specified) has counts that do (correctly) trigger pruning, but its more general counterpart does not, because the behaviour of other lexemes with the same tag is rather different (some of them giving rise to good occurrences, which make G non-zero and push the estimate above the threshold). How, therefore, should we decide how general a set of bigrams to consider when returning G and B counts for a datum encountered at run time?

We solve this problem by effectively considering the possible "views" of a datum implied by each of a range of different abstractions. For bigrams (C_1, T_1, C_2, T_2), we generalize over the left-hand and/or right hand classes, to give the three patterns $(*, T_1, C_2, T_2)$, $(C_1, T_1, *, T_2)$ and $(*, T_1, *, T_2)$. We do not generalize over tags, because there is no particular reason why edges for the same word but different syntactic categories should behave similarly (as would be implied by a pattern like $(C_1, *, C_2, T_2)$), and because generalizing over both class and tag in the same position (e.g. $(*, *, C_2, T_2)$) would be equivalent to reducing to tree-grams, which are a detailed kind of unigram.

For tree-grams themselves, which have six places, we generalize in 13 of the 63 (i.e. $2^6 - 1$) conceivable different ways, those again being the ones which seem likely to yield useful patterns.

At run time, instead of calculating an edge probability by minimizing over only three types of discriminant, we effectively minimize over the original, specific, types, and all those created by generalization as well. In fact, it is possible to precompile most of this minimization by only storing event counts that can contribute minimum values for some data that may occur at run time. Thus for the "you M" example above, the bigram $(*, np, M, character)$, for any NP, not just "you", followed by the character "M", is the one with the most informative score, and this is the only matching one whose score is retained. We also include "inhibitory" records in the data used for pruning, for cases where the counts for more specific data include enough good occurrences, and few enough bad ones, to override the pruning decision that would be implied by more general types.

The data used for pruning at run-time in the English ATIS system consists of about 4,000 records at each of the first two levels, which with suitable indexing allows reasonably efficient pruning. At each level, about three quarters of the records are generalized, and around 10% are inhibitory. This relatively small set of records is derived from around 40,000 different tree-grams and bigrams extracted during training *before* generalization. The reduction in numbers is due partly to the fact that many of these items have counts that in fact do not justify pruning, and so there is no reason to keep them; and partly to the fact that when a generalization is found that has counts that would trigger pruning, this usually allows most or all of the specific records contributing to it to be discarded.

Complication 3: Acoustic Scores

When recognizer output in the form of N-best lists is being processed, it is desirable to allow the pruning decision to be swayed by the acoustic score of the edge(s) involved. In order to do this, when we calculate probability estimates using formula 6.7, we pretend that the sample of G good occurrences on which the estimate is based came not from an overall sample of $G + B$ occurrences but of $(G + B)/\alpha$, where α is an estimate, derived from training on N-best lists, of the probability that a datum covering the given number of adjacent words with a given (maximal) acoustic score shortfall is in fact part of the correct word sequence (as defined by the reference version provided with all ATIS utterances).[2] When the word(s) involved are part of the top hypothesis in the N-best list, with a shortfall of zero, α will be close to 1, reflecting the fact that words in the top hypothesis are usually correct; for larger shortfalls, and to a lesser extent for larger numbers of words forming a sequence, α will be smaller.

Penalizing acoustically poor edges by multiplying linguistic (tree-gram and N-gram based) and acoustic scores together, rather than taking their minimum, corresponds, as pointed out in Section 6.2.1 above, to assuming statistical independence between these sources of information. This seems reasonable, because there is no obvious reason why particular *syntactic* patterns should be more characteristic of some positions in the

[2]We do not take the more obvious step of multiplying G by α instead of dividing $G + B$ by it, because G may be zero.

N-best list than others. It is also practically appropriate: if we take minima, we run the risk of finding ourselves in a situation where two edges of the same class and tag (say, two city names) but different acoustic quality both get a linguistic score that is worse than either of their acoustic scores and justifies pruning, and hence get the same overall minimum score. However, we suppose, both edges cannot be pruned, because to do so would destroy the connectivity of the chart. Clearly, in this case, we want to prune the acoustically poorer edge; but that can only happen if we multiply, rather than take minima.

A similar adjustment should in principle be made during training; if a given datum only ever occurs in acoustically poor hypotheses, it may fail ever to be correct simply because it involves words that were not uttered, rather than because it is linguistically implausible. We do not make any such adjustment, partly because it would complicate the generalization procedure and other parts of the training process, but also because if, as already argued, syntactic and acoustic scores are likely to be largely independent, then the required adjustment would not in any case make very much difference. One might even argue that if a datum involves a word sequence that is always found to be wrong in training, it *should* be penalized at run-time in exactly the way that we do by *not* adjusting in the way described.

6.2.4 Relation to other pruning methods

As the example presented at the beginning of Section 6.2 and the experiments to be described below in Section 6.5 suggest, judicious pruning of the chart at appropriate points can greatly restrict the search space and speed up processing. Our method has points of similarity with some recent work in Constraint Grammar[3] and is an alternative to several other, related schemes.

Firstly, a remarked earlier, it generalizes *tagging*: it not only adjudicates between possible labels for the same word, but can also use the existence of a constituent over one span of the chart as justification for pruning another constituent over another span, normally a subsumed one, as in the "D L" example. This is especially true in the second stage of pruning, when many constituents of different lengths have been created. Furthermore, it applies equally well to lattices, rather than strings, of words, and can take account of acoustic plausibility as well as syntactic considerations.

Secondly, our method is related to *beam search* (Woods, 1985). In beam search, incomplete parses of an utterance are pruned or discarded when, on some criterion, they are significantly less plausible than other, competing parses. This pruning is fully interleaved with the parsing process. In contrast, our pruning takes place only at certain points: currently before parsing begins, and between the phrasal and full parsing stages. Potentially, as with any generate-and-test algorithm, this can mean efficiency is reduced: some paths will be explored that could in principle be pruned earlier. However, as the results in section 6.5 below will show, this is not in practice a serious problem, because the second pruning phase greatly reduces the search space in preparation for the potentially inefficient full parsing phase. Our method has the advantage,

[3]Christer Samuelsson, personal communication, 8th April 1996; see Karlsson *et al (1995) for background.*

compared to beam search, that there is no need for any particular search order to be followed; when pruning takes place, all constituents that could have been found at the stage in question are guaranteed already to exist.

Thirdly, our method is a generalization of the strategy employed by McCord (1993). McCord interleaved parsing with pruning in the same way as us, but only compared constituents over the same span and with the same major category. Our comparisons are more global and therefore can result in more effective pruning.

6.3 Grammar specialization

As described in Section 5.2 above, the non-phrasal grammar rules are subjected to two phases of processing. In the first, "EBL learning" phase, a parsed training corpus is used to identify "chunks" of rules, which are combined by the EBL algorithm into single macro-rules. In the second phase, the resulting set of "chunked" rules is converted into LR table form, using the method of Samuelsson (1994a).

There are two main parameters that can be adjusted in the EBL learning phase. Most simply, there is the size of the training corpus; a larger training corpus means a smaller loss of coverage due to grammar specialization. (Recall that grammar specialization in general trades coverage for speed). Secondly, there is the question of how to select the rule-chunks that will be turned into macro-rules. At one limit, the whole parse-tree for each training example is turned into a single rule, resulting in a specialized grammar all of whose derivations are completely "flat". These grammars can be parsed extremely quickly, but the coverage loss is in practice unacceptably high, even with very large training corpora. At the opposite extreme, each rule-chunk consists of a single rule-application; this yields a specialized grammar identical to the original one. The challenge is to find an intermediate solution, which specializes the grammar non-trivially without losing too much coverage.

Several attempts to find good "chunking criteria" are described in the papers by Rayner and Samuelsson quoted above. In Rayner and Samuelsson (1994), a simple scheme is given, which creates rules corresponding to four possible units: full utterances, recursive NPs, PPs, and non-recursive NPs. A more elaborate scheme is given in Samuelsson (1994b), where the "chunking criteria" are learned automatically by an entropy-minimization method; the results, however, do not appear to improve on the earlier ones. In both cases, the coverage loss due to grammar specialization was about 10 to 12% using training corpora with about 5,000 examples. In practice, this is still unacceptably high for most applications.

Our current scheme is an extension of the one from Rayner and Samuelsson (1994), where the rule-chunks are trees of non-phrasal rules whose roots and leaves are categories of the following possible types: full utterances, utterance units, imperative VPs, NPs, relative clauses, VP modifiers and PPs. The resulting specialized grammars are forced to be non-recursive, with derivations being a maximum of six levels deep. This is enforced by imposing the following dominance hierarchy between the possible categories:

```
utterance > utterance_unit > imperative_VP
```

```
> NP > {rel, VP_modifier} > PP
```

The precise definition of the rule-chunking criteria is quite simple, and is reproduced in the appendix.

Note that only the non-phrasal rules are used as input to the chunks from which the specialized grammar rules are constructed. This has two important advantages. Firstly, since all the phrasal rules are excluded from the specialization process, the coverage loss associated with missing combinations of phrasal rules is eliminated. As the experiments in the next section show, the resulting improvement is quite substantial. Secondly, and possibly even more importantly, the number of specialized rules produced by a given training corpus is approximately halved. The most immediate consequence is that much larger training corpora can be used before the specialized grammars produced become too large to be handled by the LR table compiler. If both phrasal and non-phrasal rules are used, we have been unable to compile tables for rules derived from training sets of over 6,000 examples (the process was killed after running for about six hours on a Sun Sparc 20/HS21, SpecINT92=131.2). Using only non-phrasal rules, compilation of the tables for a 15,000 example training set required less than two CPU-hours on the same machine.

6.4 Discriminant-Based QLF Preferences

The supervised training process results in a database of discriminant occurrences. For each discriminant we have a count of its "good" and "bad" occurrences. We have already seen how these counts are used to provide probability estimates to drive the pruning process. But how can they be reliably used to choose between full analyses once analysis is complete?

We follow the general approach of Yarowsky (1994) and, in both pruning and QLF preference application, consider discriminants in order of their strength, ruling out options until we have exactly one analysis left.

6.4.1 Discriminant Scoring for Analysis Choice

How should the information available for the different QLFs be used to choose a single correct QLF? While many schemes are possible (several are discussed by Alshawi and Carter, 1994), we motivate ours in the following way. At any stage in applying discriminants to prefer one analysis over the others, we have a number of analyses remaining, each of which corresponds to a different set of discriminants that have yet to be applied. Our next step will be to choose one of these discriminants and throw away the analyses it doesn"t apply to (or, if in training it reliably characterized incorrect analyses, throw away the ones it *does* apply to). We want this step to be as safe as possible: that is, to select the discriminant that minimizes the chance that we will throw out the correct analysis.

Therefore, for each discriminant occurring in training, we count the number of sentences for which it is a "good" discriminant (applies to the correct analyses and, because it is a discriminant rather than just any property, not to *all* of the incorrect

ones) and the number for which it is "bad" (applying to some or all incorrect ones but not the correct one). Its run-time value is then a smoothed estimate (using the formula 6.7 derived in Section 6.2.3) of the probability that a new occurrence of it as a discriminant will be good; the closer this value is to zero or one, the stronger the discriminant is deemed to be.

In our example, "List all flights leaving Denver between eight p m and nine p m", a triple-based discriminant for the pattern "(list/show) -between ... and ..." was judged bad on 50 occasions during training and never judged good ($G = 0, B = 50$, giving an estimate of 1/2652). This makes it the strongest discriminant (the one whose probability estimate is closest to zero or one; in this case, it is close to zero), so it is applied first, and we discard the two analyses it applies to. A further strongly negative discriminant, corresponding to the construction "(list/show) while *VP*-ing", which applies to interpretations that can be paraphrased "List all flights while you're leaving Denver ...", was the next strongest, having $G = 0, B = 23$, and an estimate of 1/600. One further analysis was removed when this was applied. Next, a triple for "flight(s) -between (time) and (time)", where, as indicated by the "-", the "between" PP attaches non-low, has $G = 0, B = 12$, and an estimate of 1/182. Applying this rules out one more analysis, leaving the correct one as the only survivor. In this example, all the discriminants used were negative ones, i.e. they had probabilities close to zero; this is fairly typical.

6.4.2 Advantages of a Discriminant Scheme

Although this way of applying training data at run time is not necessarily more accurate than other schemes for a given corpus and set of judgments, it has some important practical advantages over more complex schemes such as that of of the SLT-1 system (Agnäs *al*, 1994; Alshawi and Carter (1994)). One advantage is that no optimization is required here; the most sophisticated mathematics involved is in calculating the probability estimates from the "good" and "bad" counts, and the results are not even very sensitive to the form of the formula used for that purpose. Furthermore, some of the individual functions required rather a lot of computing time, and a discriminant-based scheme allows most of them to be dispensed with.

More importantly, however, using discriminants directly makes it easy to detect and repair training errors. When the weighted sum making up the score of an incorrect analysis exceeds that for the correct one in the Alshawi and Carter scheme, it is hard to tell what specific thing, if any, has gone wrong in training; usually the error can at best be tracked to one particular preference function, and then one can only observe that if the weights had been different, another choice would have been made. In our scheme, however, the discriminants applying to the selected and the correct analyses can be compared. There will be a particular point at which the correct analysis is discarded and the incorrect one kept, and here we can diagnose the problem as follows.

Occasionally, an error is due to the necessarily discriminants not having been extracted from the analyses during training; when this occurs, we extend the code that extracts discriminants and redo the automatic part of the training (using the TreeBanker to merge the old discriminant values with the new sets of discriminants, as indicated earlier). An example of this is that initially, we did not distinguish triples resulting from

PPs attaching low from other attachments; when we began to do so, the discriminants for such triples became much more reliable.

More frequently, though, all the required discriminants are present, but one or more of those involved in the choice have unlikely-looking scores. In that case it is straightforward to find which training sentences have contributed to these scores, and to determine whether there is a problem in the code that extracts the scores from the judged data, or whether the user has misjudged some of the training sentences. In the latter case, the TreeBanker can be used to extract all sentences involving the problematic discriminant(s), present them for re-judging, and integrate them back into the database.

Thus, over time, this iterative process results in increasingly high-quality discriminants, judgments and extraction code; the errors the system makes direct developers" attention to problem areas much more easily specifically than more mathematically sophisticated schemes do, and the TreeBanker supports the rejudging that is required.

6.4.3 Numerical Metrics

Although the functionality of most of the preference functions from the SLT-1 system has been taken over by the wider range of discriminants applied directly in SLT-2, some are retained. In the English SLT system, we use four functions. Three of them penalize particular linguistic phenomena; these are, respectively, bare singular noun phrases, subject-predicate disagreements with copular "be", and high attachments of modifiers to VPs. The fourth returns the acoustic cost of the analysis, defined as the maximum acoustic cost (i.e. shortfall from the score of the acoustically top sentence hypothesis provided by Decipher) of any word in the lattice used to make it up.

One is then faced with the problem of integrating their results into the overall scheme: how can a score returned by a function be compared with the probability estimate provided by a single discriminant?

Our basic technique is to treat the return of a particular value by a particular function as a discriminant like any other. Thus, for example, if the copula-disagreement function returns a count of one on 30 occasions, and this acts as a good discriminant once and a bad one 29 times, then that function returning that value will receive a discriminant score (i.e. a correctness probability estimate, quite distinct from the value of the function) close to 1/30, just as a triple-based discriminant would for the same counts.

Two modifications are applied to this basic idea to increase the power of function-based discriminants. Both are based on the fact that for all the functions we use, larger counts should score worse, since the objects counted are signs that the QLF concerned should be dispreferred.

Sparseness

One problem is that of sparseness: a copula disagreement count of three or more is very rare, leading to a fairly weak probability estimate, simply because there unlikely to be as many as three copular verbs in any one sentence. For a given function F and value N, then, instead of using the counts for exactly the event $F(q) = N$ to estimate the likelihood of a QLF q with this count being correct, we use the sums of the counts

N	G	B
0	110	1
1	1	106
2	0	3
3	0	1

Table 6.1: Discriminant counts for a numerical preference function

for $F(qlf) = M, N_1 \leq M \leq N_2 \leq N$, where N_1 and N_2 are chosen to minimize the value of the estimate. Thus if we had the good and bad counts shown in Table 6.1, then for $N = 3$, setting N_1 to 1 and N_2 to 3 would give a revised $G = 1 + 0 + 0 = 1, B = 106 + 3 + 1 = 110$, which minimizes the value of formula 6.7 over the allowed N_1 and N_2 values.

Minimality

Sometimes, especially for longer sentences, all QLFs exhibit at least one of the phenomena which a given function is intended to penalize. Intuitively, a QLF q for which $F(q) = 2$, say, is much more likely to be correct if there is no QLF q' for which $F(q') < 2$. If we mix together the "good" and "bad" counts for occurrences of $F(q) = 2$ from sentences where 2 is the minimal value of F with those from sentences where it is not, we are likely to lose important information. This is important for all the linguistic numerical metrics; it does not so much affect the acoustic score metric, for which there is always a word sequence (and usually one or more QLFs) with zero score.

We therefore separate minimal from non-minimal scores, so that the event of function F returning value N, where N is the minimal value of F over QLFs for the sentence in question, is treated as a completely different discriminant from F returning N where N is not minimal. When this is done, the minimal/non-minimal distinction often contributes more information than any numerical differences themselves; for example, the discriminant score for the copula-disagreement metric returning a non-minimal value of 1 is far closer to the score for it returning a non-minimal value of 2 than to that for it returning a minimal value of 1.

6.5 Experiments

This section describes a number of experiments carried out to test the utility of the theoretical ideas on pruning and grammar specialization presented above. The basic corpus used was a set of 16,000 utterances from the Air Travel Planning (ATIS; Hemphill *et al.*, 1990) domain. All of these utterances were available in text form; 15,000 of them were used for training, with 1,000 held out for test purposes. Care was taken to ensure not just that the utterances themselves, but also the *speakers* of the utterances were disjoint between test and training data; as pointed out in Rayner *et al.* (1994a), failure to

Examples	Old scheme		New scheme	
	Rules	Loss	Rules	Loss
100	100	47.8%	69	35.5%
250	181	37.6%	126	21.8%
500	281	27.6%	180	14.7%
1000	432	22.7%	249	10.8%
3000	839	14.9%	455	7.8%
5000	1101	11.2%	585	6.6%
7000	1292	10.4%	668	6.0%
11000	1550	9.8%	808	5.8%
15000	1819	8.7%	937	5.0%

Table 6.2: EBL rules and EBL coverage loss against number of training examples

observe these precautions can result in substantial spurious improvements in test data results.

The 16,000 sentence corpus was analysed by the SRI Core Language Engine (Alshawi, 1992), using a lexicon extended to cover the ATIS domain (Rayner, 1994). All possible grammatical analyses of each utterance were recorded, and an interactive tool was used to allow a human judge to identify the correct and incorrect readings of each utterance. The judge was a first-year undergraduate student with a good knowledge of linguistics but no prior experience with the system; the process of judging the corpus took about two and a half person-months. The input to the EBL-based grammar-specialization process was limited to readings of corpus utterances that had been judged correct. When utterances had more than one correct reading, a preference heuristic was used to select the most plausible one.

Two sets of experiments were performed. In the first, increasingly large portions of the training set were used to train specialized grammars. The coverage loss due to grammar specialization was then measured on the 1,000 utterance test set. The experiment was carried out using both the chunking criteria from Rayner and Samuelsson (1994) (the "Old" scheme), and the chunking criteria described in Section 6.3 above (the "New" scheme). The results are presented in Table 6.5.

The second set of experiments tested more directly the effect of constituent pruning[4] and grammar specialization on the Spoken Language Translator's speed and coverage; in particular, coverage was measured on the real task of translating English into Swedish, rather than the artificial one of producing a correct QLF analysis. To this end, the first 500 test-set utterances were presented in the form of speech hypothesis lattices derived by aligning and conflating the top five sentence strings produced by a version of the DECIPHER (TM) recognizer (Murviet *et al.*, 1993). The lattices were analysed by four different versions of the parser, exploring the different combinations of turning constituent pruning on or off, and specialized versus unspecialized grammars. The

[4]We report results for the slightly earlier version of the pruner reported by Rayner and Carter, 1996, rather than the one described in this chapter. However, our experience of using the system suggests that the newer pruner gives equally good results.

	E– P–	E+ P–	E– P+	E+ P+
Morph/lex lookup	0.53	0.54	0.54	0.49
Phrasal parsing	0.27	0.28	0.14	0.14
Pruning	–	–	0.57	0.56
Full parsing	12.42	2.61	3.04	0.26
Preferences	3.63	1.57	1.27	0.41
TOTAL	16.85	5.00	5.57	1.86

Table 6.3: Breakdown of average time spent on each processing phase for each type of processing (seconds per utterance)

specialized grammar used the "New" scheme, and had been trained on the full training set. Utterances which took more than 90 CPU seconds to process were timed out and counted as failures.

The four sets of outputs from the parser were then translated into Swedish by the SLT transfer and generation mechanism (Agnäs *et al.*, 1994). Finally, the four sets of candidate translations were pairwise compared in the cases where differing translations had been produced. We have found this to be an effective way of evaluating system performance. Although people differ widely in their judgements of whether a given translation can be regarded as "acceptable", it is in most cases surprisingly easy to say which of two possible translations is preferable. The last two tables summarize the results. Table 2 gives the average processing times per input lattice for each type of processing (times measured running SICStus Prolog 3#3 on a SUN Sparc 20/HS21), showing how the time is divided between the various processing phases. Table 3 shows the relative scores of the four parsing variants, measured according to the "preferable translation" criterion.

6.6 Conclusions and further directions

Table 2 indicates that EBL and pruning each make processing about three times faster; the combination of both gives a factor of about nine. In fact, as the detailed breakdown shows, even this underestimates the effect on the main parsing phase: when both pruning and EBL are operating, processing times for other components (morphology, pruning and preferences) become the dominant ones. As we have so far expended little effort on optimizing these phases of processing, it is reasonable to expect substantial further gains to be possible.

Even more interestingly, Table 3 shows that real system performance, in terms of producing a good translation, is significantly *improved* by pruning, and is not degraded by grammar specialization. (The slight improvement in coverage with EBL on is not statistically significant). Our interpretation of these results is that the technical loss of grammar coverage due to the specialization and pruning processes is more than counterbalanced by two positive effects. Firstly, fewer utterances time out due to slow

	E– P–	E+ P–	E– P+	E+ P+
E–/P–		12–24	25–63	24–65
E+/P–	24–12		31–50	26–47
E–/P+	63–25	50–31		5–8
E+/P+	65–24	47–26	8–5	

Table 6.4: Comparison between translation results on the four different analysis alternatives, measured on the 500-utterance test set. The entry for a given row and column holds two figures, showing respectively the number of examples where the "row" variant produced a better translation than the "column" variant and the number where it produced a worse one. Thus for example "EBL+/pruning+" was better than "EBL–/pruning–" on 65 examples, and worse on 24.

processing; secondly, the reduced space of possible analyses means that the problem of selecting between different possible analyses of a given utterance becomes easier.

To sum up, the methods presented here demonstrate that it is possible to use the combined pruning and grammar specialization method to speed up the whole analysis phase by nearly an order of magnitude, without incurring any real penalty in the form of reduced coverage. We find this an exciting and significant result, and are further continuing our research in this area during the coming year. In the last two paragraphs we sketch some ongoing work.

All the results presented above pertain to English only. The first topic we have been investigating is the application of the methods described here to processing of other languages. Preliminary experiments we have carried out on the Swedish version of the CLE (Gambäck and Rayner, 1992) have been encouraging; using exactly the same pruning methods and EBL chunking criteria as for English, we obtain comparable speed-ups. The loss of coverage due to grammar specialization also appears comparable, though we have not yet had time to do the work needed to verify this properly. We intend to do so soon, and also to repeat the experiments on the French version of the CLE (Rayner *et al.*, 1996).

The second topic is a more radical departure, and can be viewed as an attempt to make interleaving of parsing and pruning the basic principle underlying the CLE's linguistic analysis process. Exploiting the "stratified" nature of the EBL-specialized grammar, we group the chunked rules by level, and apply them one level at a time, starting at the bottom. After each level, constituent pruning is used to eliminate unlikely constituents. The intent is to achieve a trainable robust parsing model, which can return a useful partial analysis when no single global analysis is found. An initial implementation exists, and is currently being tested; preliminary results here are also very positive. We expect to be able to report on this work more fully in the near future.

Appendix: definition of the "New" chunking rules

This appendix defines the "New" chunking rules referred to in Sections 6.3 and 6.5. There are seven types of non-phrasal constituent in the specialised grammar. We start by describing each type of constituent through examples.

Utterance: The top category.

Utterance_unit: `Utterance_ unit`s are minimal syntactic units capable of standing on their own: for example, declarative clauses, questions, NPs and PPs. Utterances may consist of more than one `utterance_unit`. The following is an `utterance` containing two `utterance_unit`s: "[Flights to Boston on Monday] [please show me the cheapest ones.]"

Imperative_VP: Since imperative verb phrases are very common in the corpus, we make them a category of their own in the specialised grammar. To generalise over possible addition of adverbials (in particular, "please" and "now"), we define the `imperative_vp` category so as to leave the adverbials outside. Thus the bracketed portion of the following utterance is an `imperative_vp`: "That's fine now [give me the fares for those flights]"

Non_phrasal_NP: All NPs which are not produced entirely by phrasal rules. The following are all `non_phrasal_NP`s: "Boston and Denver", "Flights on Sunday morning", "Cheapest fare from Boston to Denver", "The meal I'd get on that flight"

Rel: Relative clauses.

VP_modifier: VPs appearing as NP postmodifiers. The bracketed portions of the following are `VP_modifiers`: "Delta flights [arriving after seven p m]" "All flights tomorrow [ordered by arrival time]"

PP: The CLE grammar treats nominal temporal adverbials, sequences of PPs, and "A to B" constructions as PPs (cf Rayner, 1994). The following are examples of PPs: "Tomorrow afternoon", "From Boston to Dallas on Friday", "Denver to San Francisco Sunday"

We can now present the precise criteria which determine the chunks of rules composed to form each type of constituent. For each type of constituent in the specialised grammar, the chunk is a subtree extracted from the derivation tree of a training example (cf Rayner and Samuelsson, 1994); we specify the roots and leaves of the relevant subtrees. The term "phrasal tree" will be used to mean a derivation tree all of whose rule-applications are phrasal rules.

Utterance: The root of the chunk is the root of the original tree. The leaves are the nodes resulting from cutting at maximal subtrees for `utterance_unit`s, `non_phrasal_np`s `pp`s, and maximal phrasal subtrees.

Utterance_unit: The root is the root of a maximal subtree for a constituent of type `utterance_unit`. The leaves are the nodes resulting from cutting at maximal subtrees for `imperative_vp`s, `np`s, and `pp`s, and maximal phrasal subtrees.

Imperative_VP: The root is the root of a maximal subtree under an application of the `S` $\rightarrow$ `VP` rule whose root is not an application of an adverbial modification rule. The leaves are the nodes resulting from cutting at maximal subtrees for `non_phrasal_np`, and `pp`, and maximal phrasal subtrees.

Non_phrasal_NP: The root is the root of a maximal non-phrasal subtree for a constituent of type `np`. The leaves are the nodes resulting from cutting at maximal subtrees for `rel`, `vp_modifier`, and `pp`, and maximal phrasal subtrees.

Rel: The root is the root of a maximal subtree for a constituent of type `rel`. The leaves are the nodes resulting from cutting at maximal subtrees for `pp`, and maximal phrasal subtrees.

VP_modifier: The root is the root of a `vp` subtree immediately dominated by an application of the `NP` $\rightarrow$ `NP VP` rule. The leaves are the nodes resulting from cutting at maximal subtrees for `pp`, and maximal phrasal subtrees.

PP: The root is the root of a maximal non-phrasal subtree for a constituent of type `pp`. The leaves are the nodes resulting from cutting at maximal phrasal subtrees.

Chapter 7

Acquisition of Linguistic Knowledge

David Carter, Robert Eklund and Ian Lewin

In this chapter we describe the support provided to make the acquisition of various kinds of linguistic knowledge semi-automatic and therefore requiring less, and/or less expert, human intervention than would otherwise be the case. First, in Section 7.1, we examine the `lexmake` tool which is used to acquire lexical entries by showing users examples of word usages from a corpus. Then in Section 7.3 we look at the TreeBanker, a graphical tool for the efficient interactive disambiguation of parsed sentences. Two other training interfaces currently exist in rudimentary form, and are described briefly in the parts of this report dealing with the run-time use of the data they help to create. These are a tool for deriving preferences for the fragments created in robust parsing (Section 12.3.3), and one for training the choices involved in non-deterministic transfer (Section 12.2.3).

7.1 The Acquisition of Lexical Entries

Lexical acquisition involves determining which words need to be added to the existing lexicon and then providing, in a machine-understandable format, descriptions of their surface forms, morphology, syntax and semantics. Unaided, this can be a highly time-consuming and error-prone task. It can be time-consuming because of the sheer volume of work. It can be especially error-prone partly because of the linguistic and system-specific expertise required of a user (the task involves examining the corpus occurrences of many words, and then coding complex machine-understandable lexical entries to represent them) and partly because this expertise is often exercised only infrequently – most new words to be added to a lexicon are just new instances of commonly occurring old ones.

7.1.1 `lexmake` Tool Description

The new lexical acquisition tool is strongly corpus-based (although interactive parts can be used without a corpus). First, the corpus of new data is searched for occurrences of words not currently known by the lexicon. The search uses a simple string matching procedure without attempting morphological analysis. Unknown words, ordered by frequency of occurrence, are automatically extracted and a file of "fill-in-the-blanks" lexical entries in CLE-readable format is generated. Each entry in the file is associated with the sentences in the corpus in which it occurs, up to some predefined limit of sentences. Secondly, the user runs the interactive tool, `lexmake` which presents a simple "point-and-click" interface for filling in the blank lexical entries. For each word, the user sees the word he is currently defining, its occurrences in the corpus and a structured menu of the different choices that he is required to make. The choices themselves are presented in a linguistically motivated format and abstract away from the details of any underlying machine-readable coding. All tool-outputs are also tool-readable so that the lexica generated by the tool can also be easily modified by it. The user can spread his lexicon development over several sessions and review and modify any earlier decisions in later sessions. The tool is not intended to be used for all lexical entries, only those describable in terms of "paradigms" which are commonly occurring patterns of categories, features and values (see Carter, 1989, for details). Such words form the vast majority of cases in adaptation to a new corpus.

The tool itself is designed to be easily reconfigurable both for new languages and for different users defining lexical entries in one language. Since different users may have different levels of linguistic expertise, it is useful to be able to tailor the tool to their particular knowledge. In the simplest case, it actually proves highly useful simply to be able to modify the prompt messages for particular users so that they can easily remind themselves, in terms they understand, what a particular choice means or what distinction they are being asked to make. Reconfigurability is achieved by following our general policy of declaratively stating linguistic descriptions separately from the required computational processing. The very same processing is used both in English and Swedish lexical development – all that differs is the language description which is processed. The lexicon tool reads both a lexicon file and a language description in order to configure itself for use. Definition of the language description itself is an expert task since it requires knowledge of the CLE encoding of the lexicon. The language description includes a hierarchy of lexical category names (and associated help strings describing them), a list of associated category features and their possible values (plus descriptive help strings) including a default value, a list of language-specific special characters allowed in lexical entries, and a list of commonly-occurring language-specific patterns of irregularity.[1] Each irregularity declaration states the morphological rule name and affixes that would be used if the derivation were regular, a help string describing the derivation, a substitution operation for generating a CLE internal for-

[1] Of course, strictly speaking irregularities by definition do not follow a pattern. By "irregularities" here, we mean forms that do not conform to the spelling rules (see Appendix A) for the current language description. In this sense, nouns such as "bok" have irregular plural forms ("böcker"), and verbs such as "springa" have an irregular supine form ("sprungit"), although in both cases there are other words in the language that follow similar patterns.

mat for a root from a string which the user is prompted for, and a further help string which prompts the user for the string. The substitution operation is used to hide CLE internal formats used in morphological derivation from the user. The user is prompted for a word in a form he can readily understand and the internal format is derived by the substitution operation. For example, in Swedish one entry declares the existence of irregular adjectives which do not add an umlaut when forming the comparative and superlative forms. The declarations include the help string “Enter the new comparative string (lägre)” prompting the user to type something familiar, namely the comparative form of the adjective. The substitution operation (which in this case deletes the terminal “re”) is then used to derive the SLT internal form. The internal form can be used to derive both comparatives and superlatives. Finally, a set of language specific help messages associated with pop-up help windows can also be declared.

7.1.2 Example

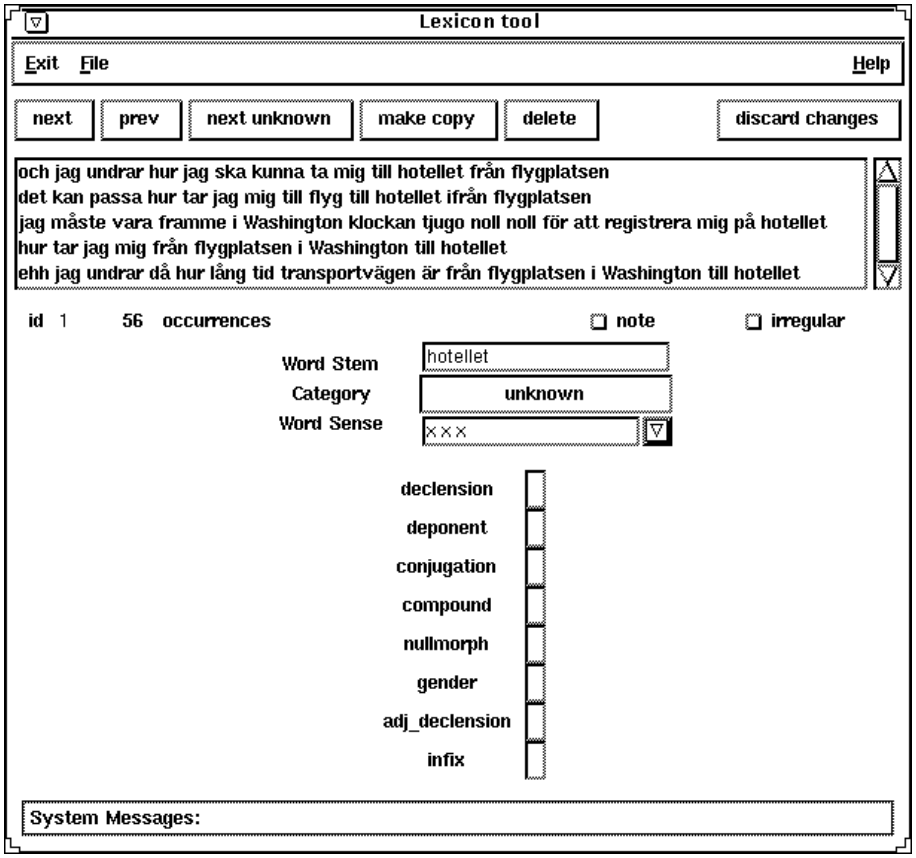


Figure 7.1: `lexmake` display for “hotellet” before any user modifications are made.

Figure 7.1 illustrates what the user sees during the definition of the lexical entry for “hotell”. At the very top of the screen are generic process and file manipulation commands, giving the ability to quit, save a file, step forwards and backwards through a file, delete an entry and so on. Below that are data associated with “hotellet”. First, five examples from the corpus in which “hotellet” appears are shown in a scrollable window. Secondly, the middle of the screen shows the word’s main properties: its stem is currently shown as “hotellet”, its category is shown as unknown and its sense is “XXX”. Since the category is unknown, no category features (declension, conjugation etc) are defined. The user’s task is to fill in the information required. First the user must define the correct word stem which in this case is “hotell” (the stems of nouns are their singular indefinite form - this information is available under “help”). By clicking on the current category “unknown”, a tree-structured menu of alternative choices such as “count noun” and “reflexive transitive verb” is posted and can be selected from. On selecting “count noun”, default modifiable values for the features declension, gender and infix are enabled. These features are the only ones valid for the count noun category. In this instance, the user needs to alter the default declension value (which is “-or(blomma)”) to “-ø(hus)” and the default gender (“common”) to “neuter”. Finally the user needs to add in a sense for “hotell”. He may either add a new one himself, such as “Härbärge”, or select from a predetermined list of common semantic class names (the very same ones used for preference functions and pruning; see Section 6.2.1). The user may also add a note, i.e. some arbitrary comment, by clicking on the note flag or enter a dialogue for adding irregular entries by clicking on the irregular flag. Once the user is happy with his entry, he may move onto the next word in the lexicon file by selecting `Next` from the top of the screen.

7.2 Swedish Usage

The current Swedish version of the lexicon tool, developed jointly by SRI and Telia, includes features deemed appropriate for people with good knowledge of Swedish but without any familiarity with the CLE or deeper linguistic knowledge. As is pointed out above, `lexmake` can easily be changed to make use of other categories that are judged more suitable for specific users. The current Swedish version includes 52 categories organised into a hierarchy. They range from the simplest possible category ‘name’ to more complex cases such as ‘reflexive intransitive verb incorporating a particle and special prepositional phrase’, as in ‘klä UT sig till (ngn/ngt)’. At the simplest level, not much grammatical knowledge is required in order to understand the categorization. The more complex the level, the more knowledge is required. All the categories are provided with Swedish examples to avoid reliance on sometimes complex grammatical terms. In the following sections, we describe the configuration of `lexmake` for the three principle categories of noun, verb and adjective.

7.2.1 Nouns

Swedish nouns are categorised according to gender, declension and compounding behaviour.

In `lexmake`, the user is presented the option of choosing ‘common’ or ‘neuter’ gender. The alternative names “n-genus” (“n-gender”) and “t-genus” (“t-gender”) are sometimes used in Swedish grammars, and one could easily display `-n` or `-t` instead, but the gender names themselves are simple enough to learn, even for naïve users.

Swedish nouns are traditionally divided into six declensions, five regular and one irregular. Irregular forms can also be divided into different paradigms that exhibit more or less regular patterns. The paradigms differ in the way plurals are formed. Instead of making the user choose between declension names or numbers – which would require that the user know the number of the declension and could pair it with the correct plural form – the plural ending of each declension is displayed, with an accompanying example word:

```
-or (blomma)
-ar (bil)
-er (färg)
-n  (äpple)
-ø  (hus)
oregelb (stad, son)
```

This way of presenting grammatical information enables even naïve users to put the nouns into the right declension paradigm.

Compounds in Swedish are formed with or without a ‘glue morph(eme)’ (‘foge-morfem’), here called ‘infixes’, for historical reasons. From a strict linguistic point of view, the glue morph(eme)s in Swedish compound formation are not true infixes, definitionally, since they do not carry meaning and occur *between*, not *inside*, morphemes. The by far most common ways to form noun–noun compounds are either without a glue morpheme or with an `-s-` glue morpheme (‘foge-s’). There are also four glue vowels (‘foge-vokaler’), and although their productivity is low, they do occur in connection with certain lemmae, such as “gata” and “kvinna”. The following list of the six compounding alternatives is presented to the user:

```
-ø (bil)
-s (fotboll)
-u (gatukök)
-a (barnatro)
-o (kvinnosak
-e (lekstugetak)
```

7.2.2 Adjectives

There are basically three adjective declensions, differing in the way they form comparative and superlative forms. Once again, the comparative, superlative and superlative attributive endings themselves with example words are presented to the user rather than using grammatical terminology.

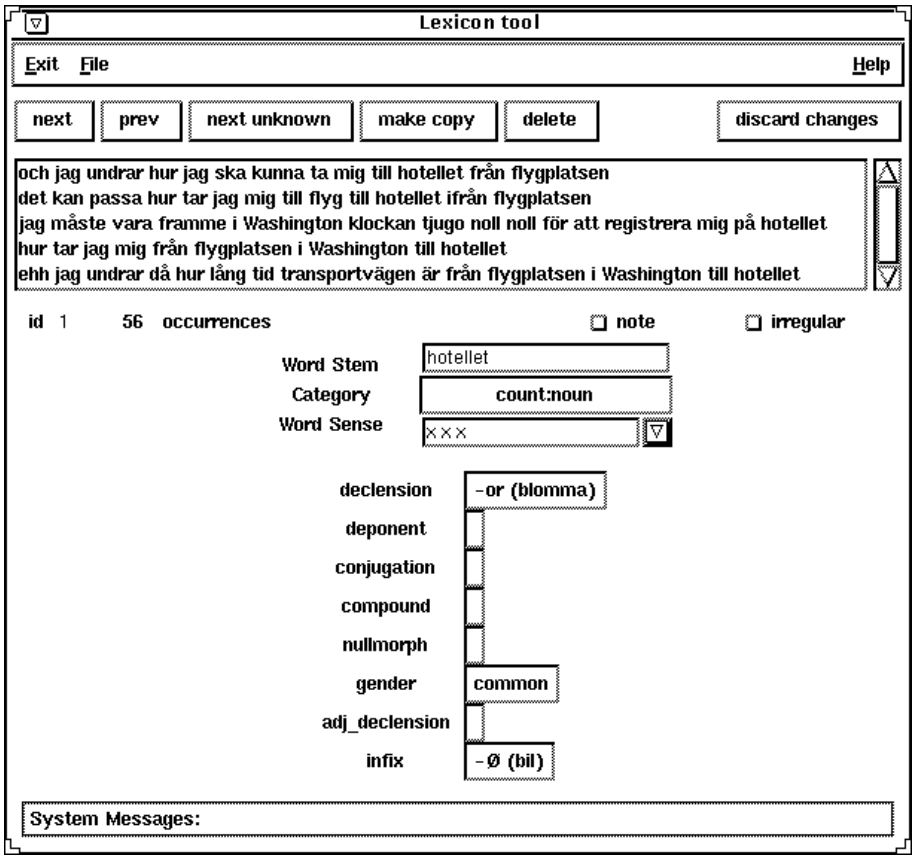


Figure 7.2: `lexmake` display for “hotellet” as a noun. “Hotellet” (“the hotel”) has been found in the corpus under scrutiny, and the user has classified it as a count noun. Default information is then automatically shown in the declension, gender and infix fields. By clicking in the said fields, menus are opened from which the user can choose the right values.

```
-are, -ast, -ast(e) (bred)
-re, -st, -sta (stor)
mer(a), mest (kul)
```

7.2.3 Verbs

Verbs in Swedish are normally described as belonging to a number of conjugations. These differ mainly according to ablaut paradigms, and could easily be categorized by naïve users through their vowel gradation, like:

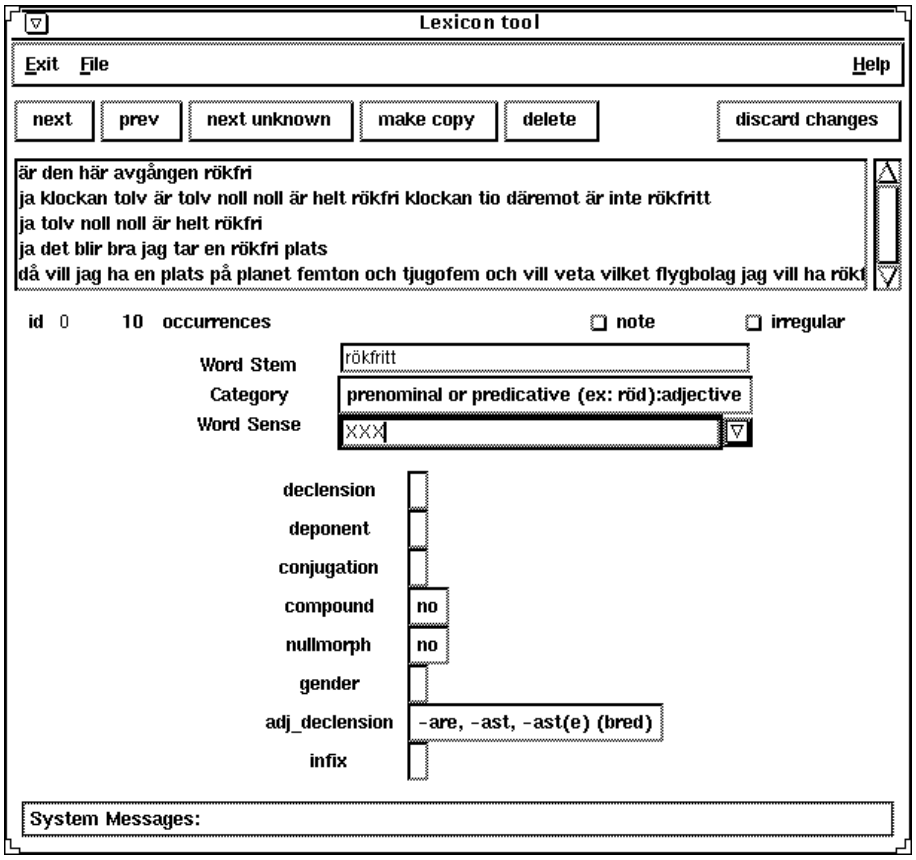


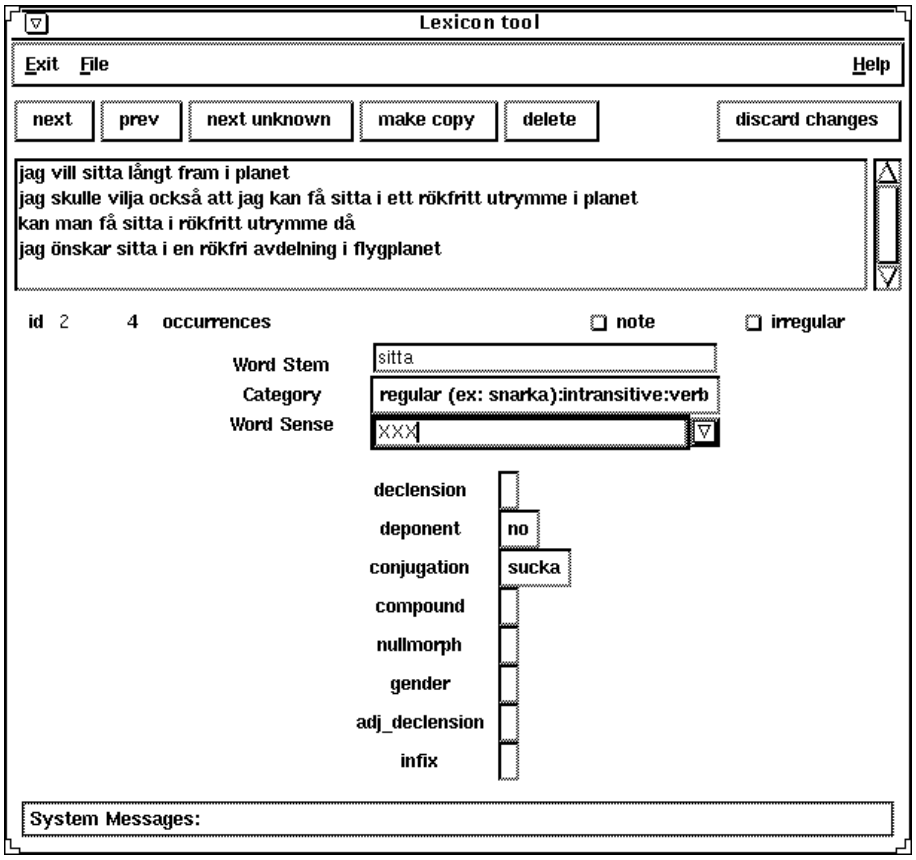
Figure 7.3: `lexmake` display for adjective. The word “rökfritt” (“non smoking”) has been found in the corpus under scrutiny, and the user has classified it as a prenominal or predicative adjective. Default information is then automatically shown in the compound, nullmorph and adjective declension fields. By clicking in the said fields, menus are opened from which the user can choose the correct values.

Example verbs	Vowel gradation
`springa - sprang - sprungit`	i – a –u
`skriva - skrev - skrivit`	i – a – i
`bjuda - bjöd - bjudit`	u/y – ö – u

... and so forth.

The present Swedish grammar, however, is not based on this traditional description, and consequently, this intuitive method is presently out of reach. This means that verbs require CLE knowledge to a larger extent than do the other word classes. This,

however, has nothing to do with the potential or functionality of `lexmake`, which easily can be adapted to suit the features of any grammar. One of the immediate goals in the development of the grammar is to alter the description of Swedish verbs towards a representation more akin to the more traditional description alluded to above. The need for doing so has very much been brought to our attention by `lexmake`, which highlighted the need for more intuitive categorical descriptions.



7.2.4 Evaluation and Conclusion

The `lexmake` tool enables a user quickly to add new lexical entries to the system through a friendly graphical interface. The interface is also designed for corpus-based development by showing corpus examples for each word to be added and naturally ordering the words to be added by their frequency of occurrence in the corpus. The tool reflects the general SLT philosophy of separating linguistic descriptions from computational processing. After an initial English version had been constructed and tested at SRI, a Swedish version was quickly and easily generated by SRI and Telia staff and successfully used for lexicon development at Telia.

Use of the graphical tool has also led us to reconsider some of our existing linguistic descriptions (e.g. verbs, described above). The previous use of a purely text-based feature-value declaration style had enabled some less linguistically natural descriptions to be developed. Representation in a graphical tool immediately highlighted the less intuitive features of the description.

7.3 The TreeBanker

This section describes the TreeBanker, a graphical tool and database management system for the supervised training involved in domain customization of the disambiguation component of the CLE. The TreeBanker presents a user, who need not be a system expert, with a range of properties (discriminants) that distinguish competing analyses for an utterance. These properties are relatively easy for a user to judge. Thus training on a corpus can be completed in far less time, and with far less expertise, than would be needed if analyses were inspected directly. We also describe how discriminants are used in choosing between QLFs at run time, and how the TreeBanker supports the detection and correction of any wrong judgments that may have led to run-time errors.

7.3.1 Motivation

In a pipelined speech understanding system such as SLT, where full, linguistically-motivated analyses of the speaker's utterances are desired, the linguistic analyser needs to generate possible semantic representations and then choose the one most likely to be correct. Even when the recognizer that provides the analyser with its input is able to deliver a unique string with reasonable confidence, the analyser faces the problems of disambiguation that are familiar from text-processing. However, the recognizer usually has to produce multiple possible word sequences because the relatively simple language models (typically N-gram based) that are efficient enough to be included in its search process are not rich enough to encode the syntactic, semantic and, perhaps, pragmatic constraints needed for full word sequence identification (Rayner *et al.*, 1994).

Therefore, if an utterance is to be correctly interpreted, the analyser needs to create the correct analysis of the correct word string, and to pick that analysis out from any others that are created along with it. Because the word-identity problem means the search space will in general be larger than for the text processing case, it is especially important that, where possible, incorrect search paths should be pruned out early on.

In practice, we can only come near to satisfying these requirements if the analyser is trained on a corpus of utterances from the same source (domain and task) as those it is intended to process. Since this needs to be done afresh for each new source, economic considerations mean it is highly desirable to do it as automatically as possible. Furthermore, those aspects that cannot be automated should as far as possible not depend on the attention of experts in the system and the representations it uses.

The TreeBanker facilitates supervised training by interacting with a non-expert user and that organizes the results of this training to provide the CLE with data in an appropriate format. The CLE uses this data to analyse speech recognizer output efficiently

and to choose accurately among the interpretations it creates. We assume here that the coverage problem has been solved to the extent that the system's grammar and lexicon license the correct analyses of utterances often enough for practical usefulness (Rayner, Bouillon and Carter, 1995).

The QLFs output by the CLE are designed to be appropriate for the inference or other processing that follows utterance analysis in whatever application the CLE is being used for. However, they are not easy for humans to work with directly in supervised training; and some way is needed to characterize the salient differences between plausible and implausible analyses because, of course, we cannot expect to encounter exactly the same QLFs at run time as during training. Both the TreeBanker and the CLE's preference mechanism therefore treat a QLF as completely characterized by its *properties*: smaller pieces of information, extracted from the QLF or the syntax tree associated with it, that are easy for humans to work with and/or are likely to be repeatedly encountered at run time. These properties are described in detail in Section 7.3.3.

7.3.2 Overview of the TreeBanker

The TreeBanker is a program for the kind of supervised training that is required to allow the CLE to distinguish correct from incorrect analyses and, as far as possible, to prune out paths likely to be incorrect. The examples we give here are all for English, but the TreeBanker has also successfully been used for Swedish and French customizations of the CLE (Gambäck and Rayner, 1992; Rayner, Carter and Bouillon, 1996). The TreeBanker takes as input the following information for each of the (typically) several thousand utterances in a corpus:

- the *reference* version of the utterance: the word string transcribed from the input speech by a human listener.
- the *recognizer output* for the utterance: the lattice or (in our case) N-best utterance list produced by the recognizer.
- the *properties* applying to the sets of QLFs produced by the analyser both for the reference version of the utterance and for the recognizer output.

The TreeBanker carries out three functions. Firstly, as detailed in section 7.3.3 below, it interacts with a user to determine the correct analysis (if any) of each sentence. The user should be familiar with the domain to which the system is being adapted and with simple linguistic concepts, but need not be a system expert. Secondly, the TreeBanker derives from the user's judgments information about the characteristics of correct and incorrect analyses, and packages it in a form which the analyser can use to select correct analyses and, when possible, prune out paths leading to incorrect ones; this process was described in Section 6.4. Thirdly, as described by Rayner and Carter (1996), it creates a library of verified analyses which are used to train the specialized grammar need for fast parsing. We conclude with a discussion of the TreeBanker's use in a particular system and application and the degree to which our goals have been achieved.

7.3.3 The Supervised Training Process

Even for an expert, inspecting all the analyses produced for a sentence is a tedious and time-consuming task. There may be dozens of analyses that are variations on a small number of largely independent themes: choices of word sense, modifier attachment, conjunction scope and so on. Further, if the representation language is designed with semantic and computational considerations in mind, there is no reason why it should be easy to read even for someone who fully understands it. And in fact, as already argued, it is preferable that selection of the correct analysis not require the involvement of an expert at all. (In practice, at the current state of development, some decisions needed by the TreeBanker are tricky enough that they have to be left for an expert to make them, but these occur in only a very small minority of sentences).

Properties and Discriminants

We have therefore taken the approach of defining a number of different types of *property*: facts about analyses that, in most cases, can be presented to non-expert users in a form they can easily understand. Those properties that hold for some analyses of a particular utterance but not for others we referred to in Section 6.2.1 as *discriminants*. Discriminants that fairly consistently hold for correct but not (some) incorrect analyses, or vice versa, are likely to be useful in distinguishing correct from incorrect analyses at run time. Thus for training on an utterance to be effective, we need to provide enough "user-friendly" discriminants to allow the user to select the correct analyses, and as many as possible "system-friendly" discriminants that, over the corpus as a whole, distinguish reliably between correct and incorrect analyses. Ideally, a discriminant will be both user-friendly and system-friendly, but this is not essential.

The TreeBanker derives properties directly from the QLFs produced by the CLE and from their associated parse trees. The database of analysed sentences that it maintains contains only these properties and not the analyses themselves. The TreeBanker presents properties to the user in a convenient graphical form, exemplified in Figure 7.4 for the sentence "Show me the flights to Boston serving a meal". The user may click on any discriminant with the left mouse button to select it as correct, or with the right button to select it as incorrect. The types of property currently extracted, ordered approximately from most to least user-friendly, are as follows; examples are taken from the six QLFs for the sentence used in figure 7.4.

- *Constituents*: ADVP for "serving a meal" (a discriminant, holding only for readings that could be paraphrased "show me the flights to Boston while you're serving a meal"); VP for "serving a meal" (holds for all readings, so not a discriminant).

- *Semantic triples*: relations between word senses mediated usually by an argument position, preposition or conjunction (Alshawi and Carter, 1994). Examples here (abstracting from senses to root word forms, which is how they are presented to the user) are "flight to Boston" and "show -to Boston" (the "-" indicates that the attachment is not a low one; this significantly affects the likelihood

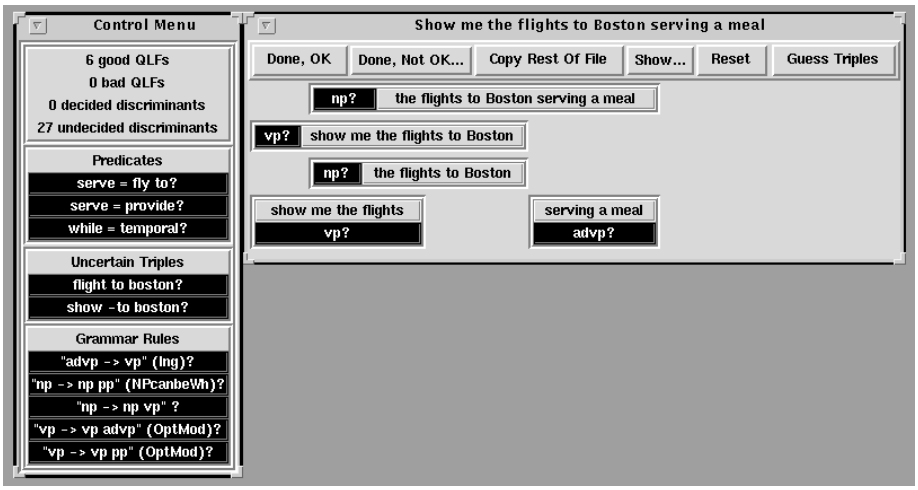


Figure 7.4: Initial TreeBanker display for “Show me the flights to Boston serving a meal”

of such discriminants being correct). Argument-position relations are less user-friendly and so are not displayed.

- *Word senses*: “serve” in the sense of “fly to” (“does United serve Dallas?”) or “provide” (“does that flight serve meals?”).
- *Sentence type*: imperative sentence in this case (other moods are possible; fragmentary sentences are displayed as “elliptical NP”, etc).
- *Grammar rules used*: the rule name is given. This is occasionally useful for experts in the minority of cases where their intervention is required.
- *Numerical metric values*: values of certain metrics applied to the code, e.g. to quantify imbalances between conjuncts. These, and following discriminant types, are not user-friendly at all, and are not displayed, but are often reliable enough for run-time use.

Two additional types of discriminant are not shown to the user but are used at run-time, for pruning rather than for QLF preferences; they are bigrams and tree-grams, as described earlier in see Section 6.2.1.

In all, 27 discriminants are created for this sentence, of which 15 are user-friendly enough to display, and a further 28 non-discriminant properties may be inspected if desired. This is far more than the three distinct differences between the analyses (“serve” as “fly to” or “provide”; “to Boston” attaching to “show” or “flights”; and,

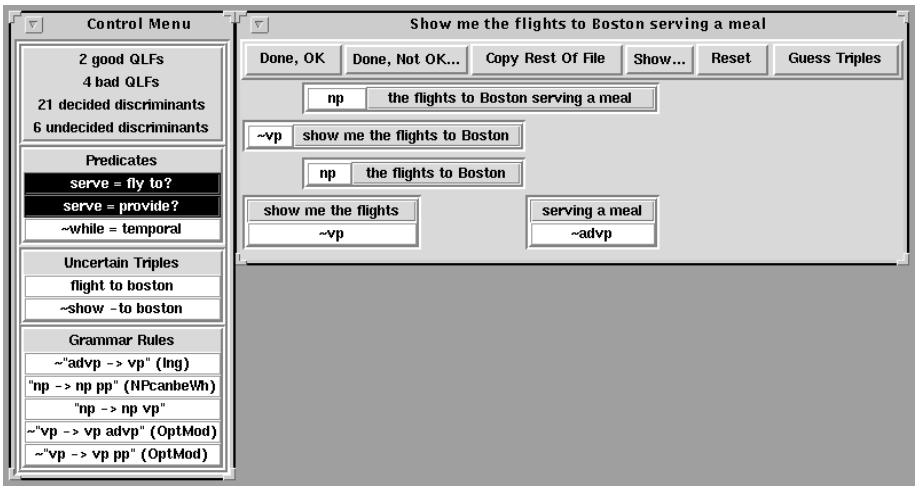


Figure 7.5: TreeBanker display after approving topmost “np” discriminant

if “to Boston” does attach to “flights”, a choice between “serving a meal” as relative or adverbial). The effect of this is that the user can give attention to whatever discriminants he[2] finds it easiest to judge; other, harder ones will typically be resolved automatically by the TreeBanker as it reasons about what combinations of discriminants apply to which analyses. Thus if the user selects “the flights to Boston serving a meal” as an NP, the TreeBanker can resolve *all* the other discriminants except the two for the sense of “serve”; and only those two remain highlighted in the display, as shown in Figure 7.5. So, for example, there is no need for the user to make the trickier decision about whether or not “serving a meal” is an adverbial phrase.

The TreeBanker’s interface often acts like this to simplify inspection of sentences whose discriminants combine to produce an otherwise unmanageably large number of QLFs. As a further example, the sentence “What is the earliest flight that has no stops from Washington to San Francisco on Friday?” yields 154 QLFs and 318 discriminants, yet the correct analysis may be obtained with only two selections. Selecting “the earliest flight ... on Friday” as an NP eliminates all but twenty of the analyses produced, and approving “that has no stops” as a relative clause eliminates eighteen of these, leaving analyses which are both correct for the purposes of translation. 152 incorrect analyses may thus be dismissed in less than fifteen seconds.

The utterance “Show me the flights serving meals on Wednesday” demonstrates the TreeBanker’s facility for presenting the user with multiple alternatives for determining correct analyses. The following decisions must be made:

[2]We offer the customary apologies for this use of pronouns, and offer the excuse that most use of the TreeBanker to date has been by men.

- Does “serving” mean “flying to” or “providing”?
- Does “on Wednesday” modify “show”, “flights”, “serving” or “meals”?
- Does “serving” modify “show” or “flights”?

but this can be done by approving and rejecting various constituents such as “the flights serving meals” and “meals on Wednesday”, or through the selection of triples such as “flight -on Wednesday”. Whichever method is utilised, the user can resolve the 14 QLFs produced for this sentence within twenty seconds.

Additional Functionality

The interactive part of the TreeBanker also supports diagnosing and categorizing coverage failures. The user may suspect that *none* of the provided analyses is correct. This situation often becomes apparent when the TreeBanker insists on automatically assigning incorrect values to some discriminants when the user makes decisions on others; the coverage failure may be confirmed, if the user is relatively accomplished, by inspecting the non-constituent properties as well and verifying that the correct parse tree is not among those offered. Then the user may mark the sentence as “Not OK” and classify it under one of a number of failure types, optionally typing a comment as well. At a later stage, a system expert may ask the TreeBanker to print out all the coverage failures of a given type as an aid to organizing work on grammar and lexicon development.

For some long sentences with many different readings, more discriminants may be displayed than will fit onto the screen at one time. In this case, the user may judge some discriminants (scrolling if necessary to find them), and ask the TreeBanker thereafter to display only *undecided* discriminants; these will rapidly reduce in number as decisions are made, and can quite soon all be viewed at once.

If the user changes his mind about a discriminant, he can click on it again, and the TreeBanker will take later judgments as superceding earlier ones, inferring other changes on that basis. Alternatively, the “Reset” button may be pressed to undo all judgments for the current sentence.

Once part of the corpus has been judged and the information extracted for run-time use (see next section), the TreeBanker may be told to resolve discriminants automatically when their values can safely be inferred. In the ATIS domain, “show -to (city)” is a triple that is practically never correct; the user can then be presented with an initial screen in which that choice, and others resulting from it, are already made. This speeds up his work, and may in fact mean that some sentences do not need to be presented at all.

In practice, coverage development tends to overlap somewhat with the judging of a corpus. In view of this, the TreeBanker includes a “merge” option which allows existing judgments applying to an old set of analyses of a sentence to be transferred to a new set that reflects a coverage change. Properties tend to be preserved much better than analyses as coverage changes; only properties, and not analyses, are kept in the corpus database, and so the vast bulk of the judgments made by the user can be preserved.

7.3.4 Evaluation and Conclusions

Using the TreeBanker it is possible for a linguistically aware non-expert to judge around 40 sentences per hour after a few days practice. When the user becomes more practised, as will be the case if he judges a corpus of thousands of sentences, this figure rises to around 170 sentences per hour in the case of our most experienced user. Thus it is reasonable to expect a corpus of 20,000 sentences to be judged in around three person weeks. A much smaller amount of time needs to be spent by experts in making judgments he felt unable to make (perhaps for one per cent of sentences once the user has got used to the system) and in checking the user's work (the TreeBanker includes a facility for picking out sentences where errors are mostly likely to have been made, by searching for discriminants with unusual values). From these figures it would seem that the TreeBanker provides a much quicker and less skill-intensive way to arrive at a disambiguated set of analyses for a corpus than the manual annotation scheme involved in creating the University of Pennsylvania Treebank; however, the TreeBanker method depends on the prior existence of a grammar for the domain in question, which is of course a non-trivial requirement.

We also plan to implement schemes such as that of Engelson and Dagan (1996) to select corpus sentences whose judging is likely to provide useful new information, rather than those that merely repeat old patterns. We expect the number of sentences needing to be judged, and hence the time required, to approximately halve when this is done.

Chapter 8

Rational development methodology

Manny Rayner

8.1 Introduction

This chapter will describe in detail the methodology we have used to develop the coverage of the rule-based parts of the system: the lexica, morphological descriptions, sets of grammar and transfer rules, and so on. To a lesser extent, the same methodology also drives the development of the other system components, in particular the engines (parsers, generators, etc.) which carry out the actual processing.

The basic idea is the usual one: take a set of example utterances (the *development set*), run the system on it, evaluate the results, and carry on iterating the edit-debug-test cycle until performance is high enough. The question we will focus on here is that of how to construct the development set. There are a number of reasonable requirements on the design of a such a set, some of which represent tradeoffs or tensions. We begin by assuming that we have a large sample of realistic domain utterances (the *main domain corpus*). In the final analysis, we are interested on improving coverage on this set.

1. The development set should be small enough that it it practically feasible to run tests on it frequently, and interpret the results easily. For this reason, it is not in general possible to use the main domain corpus as the development set.

2. The development set should be large enough that it contains examples of all the important (i.e. frequent) problems in the full corpus.

3. The relative importance of two given sentences from the development set should be clear. Thus if sentence 1 contains only instances of common words and constructions, while sentence 2 contains at least some unusual elements, then failure on sentence 1 should be more serious than failure on sentence 2.

4. It should be as easy as possible to extract the development set from the main domain corpus.

In summary, we want some fairly easy way to derive a smallish set of utterances from the main domain corpus, and be confident that if we do well on the small set then we will do at least reasonably on the original one. We will call a set of this kind a *representative corpus*.

An early methodology for building representative corpora was described in the SLT-1 report. In essence, the grammar was used to categorize the different types of utterance and extract the common patterns. This was useful for systematically improving performance of the transfer component, but presupposed that grammatical coverage of the source language was already adequate. In fact, by its very construction, the representative corpus only contained utterances which were assigned an analysis by the grammar, and was thus unsuitable for grammar development.

In the SLT-2 project, we started work on the Swedish system at a lower point. The existing Swedish grammar failed to deliver good coverage of the Swedish corpora, and our first goal was to achieve this. These constraints dictated a simpler, less elegant, but more robust approach to constructing representative corpora, which works in a bottom-up mode. The rest of this chapter will provide a full description of the method. We will illustrate with examples taken from the Swedish "Wizard of Oz" (SWoZ) corpus (see Chapter 2).

8.2 Constructing representative corpora

Since translation works bottom-up (see Section 1.2), it is unnecessary to attempt to extend the grammar and other rule-sets to the point where all utterances can be parsed as complete units. Thus the first step in the process of constructing the representative corpus is manually to split up the utterances in the main domain corpus into suitable segments, each of which can feasibly be translated as an independent unit. We refer to the result as the *split corpus*. The vertical bars in the following example utterances from the SWoZ corpus show how they are segmented in the split corpus:

```
resa stockholm berlin | okej | den andra i sjätte
(trip stockholm berlin | okay | june second)

ehh | vad kostar den biljetten | är det den billigaste
biljetten
(uh | what does that ticket cost | is it the cheapest
ticket)

sexton och femtio den andra juni | hur dags är jag
framme i berlin
(sixteen fifty on june second | when do I arrive in
berlin)

ja | sexton noll noll | vilken flygplats landar det
```

```
på i paris
(yes | sixteen hundred | at which airport do I land
in paris)

jaha | då bokar jag tolv noll fem | går den direkt
(okay | then I'll take twelve oh five | is it direct)

ehh | nej | det är för tidigt | något senare
(uh | no | that's too early | something later)
```

The next step is to extend the lexicon to achieve reasonable basic coverage of the domain at the lexical level. We do this using the `lexmake` lexicon acquisition tool (see Section 7.1). We then extend the grammar to achieve some level of non-trivial grammatical coverage on the split corpus. Coverage doesn't need to be terribly good for the subsequent steps to work; our experience suggests that being able to get analyses for about half the utterances is enough. Once the initial lexicon and grammar are in place, we create and tag a treebank (see Section 7.3), and derive a first set of pruning data (Section 6.2).

We can now use the initial phases of linguistic analysis as a part-of-speech tagger, by analyzing up to and including the lexical pruning phase and then extracting the best sequence of lexical edges. Tags consist of the major category symbol only, e.g. "noun" and "verb"; more fine-grained distinctions are discarded. We perform this type of tagging on the split corpus, and call the result the *tagged corpus*. We then group utterances from the tagged corpus into equivalence classes under the relationship of having the same tag-sequence.

The next step is manually to regroup the classes produced by the previous step where necessary. In some cases, this involves reclassifying utterances which were incorrectly tagged; in others, a group may be split into two or three smaller groups, if the relevant utterances are intuitively dissimilar enough. For instance, the following two utterances receive the same tag-sequence, since `flyger` ("fly") and `finns` ("there are") are both tagged as `v`. However, the difference between an intransitive and an existential verb is great enough that it seems reasonable to assign the utterances to different groups.

```
vilka bolag flyger från dallas till denver
(which companies fly from dallas to denver)

vilka turer finns från boston till oakland
(which flights are there from boston to oakland)
```

The "regrouping" step can be performed by non-experts at the rate of several thousand sentences a day, using a text editor, once the corpus has been formatted to facilitate this. For each of the new classes, we also manually designate an element which intuitively is "most typical" of the class. This step can also be performed quickly by non-experts using an editor.

Finally, we construct the representative corpus by selecting the designated element from each class, and order the results by the size of the classes represented. In view of

the bottom-up nature of processing, we have found it most meaningful to define the size of a class as the total number of words represented, as opposed to counting utterances.

We conclude the example by presenting an initial segment of the representative corpus derived from the SWoZ corpus. The actual corpus file lists for each representative utterance the full set of utterances which it represents; here, we have suppressed this extra information in the interests of brevity. The format of each record in the file is

```
rep_sent(<N>,<Utterance>).
```

where `<Utterance>` is the representative sentence, and `<N>` is the total number of words in the group of similar utterances which it represents.

```
rep_sent(423,'tack så mycket').
(thank you very much)

rep_sent(284,'urban kalle trea femma').
(u k three five)

rep_sent(279,tack).
(thank you)

rep_sent(209,ja).
(yes)

rep_sent(177,'den sjätte maj').
(may seventh)

rep_sent(161,och).
(and)

rep_sent(157,'jag vill boka en resa från washington till
chicago den sextonde maj').
(i want to book a flight from washington to chicago on may
sixteenth)

rep_sent(151,'går planet direkt till new york').
(does the plane fly direct to new york)

rep_sent(130,'jag skulle vilja boka en resa från washington
till chicago den sextonde maj').
(i would like to book a flight from washington to chicago
on may sixteenth)

rep_sent(126,'då bokar jag den resan').
(then i'll book that flight)

rep_sent(125,'jag skulle vilja boka en resa från chicago
```

till stockholm’).
(i would like to book a flight from chicago to stockholm)

rep_sent(120,’när är jag framme i stockholm’).
(when do i arrive in stockholm)

rep_sent(109,’jag vill boka en flygresa från nice till stockholm’).
(i want to book a flight from nice to stockholm)

rep_sent(105,okej).
(okay)

rep_sent(104,’då tar vi den’).
(i’ll take it then)

rep_sent(102,’det låter bra’).
(that sounds fine)

rep_sent(102,’jag vill åka från new york till boston den tionde maj’).
(i want to fly from new york to boston on may tenth)

rep_sent(100,’jag skulle vilja åka från nice till stockholm den fjortonde juni’).
(i would like to fly from nice to stockholm on june fourteenth)

rep_sent(92,’hej då’).
(goodbye)

rep_sent(91,’då vill jag boka den resan’).
(i’ll book that flight then)

rep_sent(89,’vilken flygplanstyp är det’).
(what type of aircraft is it)

rep_sent(80,’tack ska du ha’).
(thank you very much)

rep_sent(78,ehh).
(uh)

rep_sent(77,’vad heter flygplatsen i paris’).
(what is the airport in paris called)

rep_sent(75,’tack så mycket för hjälpen’).
(that you very much for your assistance)

rep_sent(73,’hur tar jag mig till hotellet från flygplatsen’).
(how do i get from the airport to the hotel)

rep_sent(72,’är det några stopp på vägen till boston’).
(are there any stops on the way to boston)

rep_sent(72,’jag vill beställa en flygresa från washington den sextonde maj till chicago’).
(i want to book a flight from washington on may sixteenth to chicago)

rep_sent(70,’vilket flygbolag reser jag med’).
(which airline am i travelling with)

rep_sent(64,’jag vill åka till washington från boston’).
(i want to fly from washington to boston)

rep_sent(63,’jag vill resa så billigt som möjligt’).
(i want to travel as cheaply as possible)

rep_sent(63,’är det en d c tia’).
(is it a d c ten)

rep_sent(63,’en resa från münchen till paris’).
(a trip from munich to paris)

rep_sent(60,’tack för det’).
(thank you)

rep_sent(60,’jag skulle vilja ha en resa till boston’).
(i would like a trip to boston)

rep_sent(58,’jag skulle vilja flyga från frankfurt till new york’).
(i would like to fly from frankfurt to new york)

rep_sent(57,’det blir bra’).
(that’s fine)

Chapter 9

English Coverage

Manny Rayner

The following is the SLT-1 report chapter on English coverage. It will be updated in the final version of this report.

9.1 Overview of English linguistic coverage

This chapter describes the coverage of the grammar and lexicon on the domain corpus. Much of the material will be presented at a level of detail which the average reader is likely to find tedious: we thus begin by explaining our reasons for going to these lengths. Basically, we want to establish two things about the principled linguistic approach we have adopted. The first is *sufficiency*: we want to show that it is capable of achieving good coverage of a real spoken-language corpus. The second is *necessity*: we also want to show that the corpus in fact displays a wide enough range of surface variation that it would pose serious problems for a grammar formalism much less elaborate than ours. This is particularly the case with regard to the structure of verb phrases, as discussed in Section 9.7.

Our presentation in this chapter will divide up analyses of types of construction into pre-theoretically meaningful "chunks": lexical items, noun phrases, verb phrases, preposition phrases, numbers, clauses and whole utterances. We will consider the constructions of each type that are observed to occur, and describe how the grammar deals with those which are currently within coverage. The main point at issue here is the extent to which special rules, specific either to the domain in particular or to spoken language in general, are required. In the final section, we consider sentences currently outside coverage, examining them to discover the extent to which they represent problems for the present framework. Here, we will mainly be interested in estimating the practical coverage limit inherent in our approach.

The bottom-line results are encouraging. The present level of coverage, on "A" and "D" class ATIS sentences of length up to 15 words, is about 91%. This has been achieved by taking the general CLE grammar and adding to it only a small number of

domain-specific and spoken-language specific grammatical rules and a set of domain-specific lexical entries, nearly all of which are regular open-class words. The arguments in the final section suggest that coverage could be increased to about 97% without major changes to the current grammar, by adding about 25 new rules and modifying about 10 to 20 existing ones in fairly well-specified ways. We estimate that these improvements could almost certainly be carried out with less than six person-months of work.

9.1.1 Non-standard aspects of the CLE grammar

Most of the CLE grammar is uncontroversial. We list here the small number of non-standard features it possesses, together with some justifications for the approaches taken.

"Ordinal determiners"

Superlative adjectives and ordinals are treated in an unusual way. Unlike other adjectives, superlatives are classed as a type of ordinal expression like *first* or *next*; then NPs containing an ordinal are given a constituent structure which makes the ordinal part of the DET. So for example the top-level constituent structure of *"the cheapest flight"* is

$$[_{NP}\ [_{DET}\ the\ cheapest]\ [_{NBAR}\ flight]]$$

The main advantages resulting from adoption of this analysis are semantic, and are relevant to processing stages not used in the SLT project. It is however worth noting some linguistic data which is at least not inconsistent with the "ordinal determiner" analysis. Determiner+superlative adjective combinations can occur free (e.g., *"Show me [the largest]"*) and can combine with *of* (e.g., *"the cheapest of the flights"*), patterns which are not possible with other adjectives, but which are automatically licensed by the "ordinal determiner" treatment. Also, superlative adjectives cannot pre-modify bare plurals, e.g., **"Cheapest flights"* while others can, which also falls out of our treatment. On the negative side, extra rules are needed for constructions like *your cheapest flight* (combination of possessive determiner and ordinal); these are not currently within coverage. We are far from certain that the "ordinal determiner" idea is fully correct, but taking everything into consideration it seems at any rate no worse than the normal account.

Post-modification of NPs

The analysis of post-modified NPs is also non-standard: for example, the top-level constituent structure of *"All flights to Boston"* would in most grammars be

$$[_{NP}\ [_{DET}\ all]\ [_{NBAR}\ flights\ to\ Boston]]]$$

The CLE grammar, however, assigns this phrase the structure

$$[_{NP}\ [_{NP}\ all\ flights]\ [_{PP}\ to\ Boston]]$$

The PP *to Boston* is in other words modifying an "inner" NP, rather than an NBAR: similar analyses obtain for NPs with VP, REL or ADJP post-modification.

We feel that our treatment of NP post-modification can be backed up by a fair amount of linguistic evidence. Firstly, there are cases of post-modification in NPs which lack a clear NBAR, e.g., *"those leaving before one p m"*, *"something after lunch"* or *"are there [any on Tuesday]"*. On a standard account, NPs like these will need several extra rules. Secondly, there are cases where extraction of the inner NP appears to take place, e.g., *"What flights do you have with first class"*: the simplest way of dealing with these must be to have *with first class* modify the NP gap associated with *what flights*. These constructions are currently not covered by the grammar (see also Section 9.9), but our treatment makes the necessary changes fairly simple to carry out.

"Big PPs"

The third important place where the grammar deviates from standard practise arises in connection with sequences of PPs. For example, the constituent structure assigned by most grammars to a VP like *"travel from Atlanta to Boston"* would be

$$[_{VP}\ [_{VP}\ [_{VP}\ travel]\ [_{PP}\ from\ Atlanta]]\ [_{PP}\ to\ Boston]]$$

The CLE grammar, in contrast, gives it the "flatter" structure

$$[_{VP}\ [_{VP}\ travel]\ [_{PP}\ from\ Atlanta\ to\ Boston]]$$

In general, sequences of PPs become constituents.

We are somewhat uncertain about the theoretical merits of the idea, but can at least point to the following reasons for preferring the "big PP" analysis:

1. Multiple PPs can be fronted, e.g., *"on flight eleven forty five on Delta Airlines how much is first class"*. This supports the view that they are constituents.

2. Multiple PPs can also occur on their own as elliptical utterances, e.g., *"in the morning on American Airlines"* or *"from Denver to Washington D C on Monday November eleventh nineteen ninety one"*.

3. The adjective *available* often appears to subcategorize for a list of one or more PPs. For example, in the sentence *"show me the flights available from Dallas to Baltimore August third"* it is plausible that *available from Dallas to Baltimore August third* is a constituent: this is easiest to achieve in the grammar if one sense of *available* takes a PP which may be multiple.

Passive gaps

The approach used to deal with passive constructions is also non-standard. For example, a sentence with a passive VP like *"What plane is used on the Continental flight"* will be analyzed as having a gap after *used* arising from the passivized object of *use*, corresponding semantically to the implicit agent. The merits of the idea are argued at length in Pulman (1987).

Apart from the four cases just discussed, the CLE grammar is standard in its treatment of most linguistic phenomena that have received attention in the literature. In the following sections, we will examine in detail its performance on various types of construction in the ATIS corpus.

9.2 Lexical items

Before looking at the grammar proper, we briefly review the lexical entries; the main point we will focus on is the extent to which the lexicon is domain-dependent. First, however, we summarize some overall frequency statistics, to provide a background for the following results. The portion of the development corpus that is currently within coverage (about 91% of a set of 4615 A and D class sentences of 15 words or less) included examples of 732 distinct lexical entries, counting different inflections of the same entry as distinct. Of these, 305 occurred more than 10 times; 122 occurred between five and 10 times inclusively; 146 occurred two to four times; and 159 occurred only once.

Deciding which entries are "domain-dependent" is unfortunately not clear-cut. Many entries are for words and phrases that are particularly common within the ATIS domain. If, however, it seems clear that they occur with reasonable frequency, used in the same way, in other contexts, then we will count them as domain-independent on the grounds that a sufficiently large general lexicon for English would include them. The decision is particularly hard to make in the case of proper nouns: somewhat arbitrarily, we have counted names of states and large cities as domain-independent, but names of airports and airlines as domain-dependent. Thus for example the following are all counted as domain-independent: *Atlanta*, *Washington D C*, *Pittsburgh Pennsylvania*, *connecting* (e.g., *"flights connecting through Denver to Oakland"*), *service* (e.g., *"what other airlines service that route"*), *first class* (e.g., *"how much is first class"*), *time zone* and *one way* (e.g., *"what are the lowest one way flights from Denver to Atlanta"*).

On this criterion there turn out to be 51 domain-specific entries, which break down as follows. 29 entries are for proper nouns, of which 17 are names of airlines (e.g., *Delta Airlines*, *U A*), 10 names of airports (e.g., *Baltimore Washington*, *D F W*), and the last two are the words *airport* and *coach* used as proper nouns. Nine more entries are common nouns referring to types of aircraft (e.g., *D C ten*, *Boeing seven thirty seven*). The remaining 13 entries are singular and plural forms of the following common noun phrases (in most cases only one form occurs): *class of service*, *seating capacity*, *round trip*, *rental car*, *car rental*, *a m flight*, *p m flight*, *coach economy class* and *international airport*. All of these could in theory have been derived compositionally from their components, but there was in each case some reason which suggested that they would be better treated as lexicalized constituents. For example, the phrase *class of service code* occurs frequently in the corpus, and it is not in general desirable to allow an NP as complex as *class of service* (read compositionally) to act as an NBAR pre-modifier. The argument is that the phrase is permissible precisely because *class of service* is lexicalized in the context of the ATIS domain. This raises interesting questions, though they will have to wait for another occasion to be discussed further.

9.3 Non-recursive NPs

We will now begin our walk through the grammar proper by looking at the non-recursive NPs, partly because these are by a considerable margin the most common type of phrase. Due to the way in which NP post-modification is handled by the grammar (see Section 9.1.1), it makes sense to split the treatment of NPs into "recursive NPs" and "non-recursive NPs". We define an NP to be recursive if at least one of its constituents is a non-lexical NP phrase; the remaining NPs are either lexical or non-recursive. We give examples and approximate frequencies for the different types of non-recursive NPs, and then describe how each type of phrase is defined by the grammar. About 1.5% of all non-recursive NPs occurring in the corpus are not currently within coverage. The majority of these are bare singular NPs typical of colloquial spoken English, which are discussed at greater length in Section 9.9.

"Basic" NPs We will refer to NPs built up from recursive application of a core set of rules for NPs, NBARs, ADJPs, DETs and ORDINALs as "basic" NPs: these account for about 74% of the non-recursive NPs in the corpus. Examples are *"fares"*, *"all flights"*, *"breakfast"*, *"Sunday night"*, *"the fare codes"*, *"the cheapest one way flights"*, *"the San Francisco limousine service"*

Time and date NPs Non-recursive NPs formed using constructions specifically associated with dates and times account for about 10% of the non-recursive NPs. Examples: *"eight o'clock"*, *"September fifteenth"*, *"seven fifty five a m"*, *"three"*, *"five forty"*, *"next Wednesday"*, *"Tuesday October first"*.

"Code" NPs Non-recursive NP constructions using "code" expressions, and often one or more of the "core" non-recursive NP rules as well, account for about 10% of the non-recursive NPs. Examples: *"American flight four eight seven"*, *"D L seven four six"*, *"flight U A one thirty"*, *"flight four seven six"*, *"fare code F N"*, *"flight number seventeen sixty five"*, *"Delta flight number seven oh nine"*.

Bare determiner NPs NPs consisting of a bare determiner constitute about 2% of the non-recursive NPs. Examples: *"any"*, *"which"*, *"one"*, *"the same"*, *"some"*, *"how many"*, *"the latest"*, *"the least expensive"*.

"kind of" NPs NPs using the "kind/sort/type of" construction account for about 2% of the non-recursive NPs. Examples: *"what type of plane"*, *"what kind of aircraft"*, *"what types of ground transportation"*, *"what sort of ground transportation"*, *"types of planes"*.

Special NPs NPs built using rules possibly invalid for a general English grammar, or added specially for the ATIS domain account for about 2% of all non-recursive NPs. Examples: *"cost"*, *"latest flight"*, *"last flight"*, *"downtown Pittsburgh"*.

9.3.1 "Basic" non-recursive NPs

Approximately 74% of the non-recursive NPs in the corpus are built up by recursive application of a core set of 16 rules: three for NPs; six for NBARs; five for DETs; and

one each for ADJPs and ORDINALs. We now describe these rules. Since the features used are by the nature of things somewhat arbitrary, we will not specify them in detail. Instead, we describe what the intended effect of the feature settings is supposed to be, in terms of licensing some desired derivations while blocking other undesired ones. We consequently present the rules in a highly schematic form.

At the top level, we have the three core NP rules,

```
NP -> DET NBAR

NP -> NBAR:[num=plur]

NP -> NBAR:[mass=y]
```

The first and second of these are unproblematic. The first makes an NP out of a determiner+NBAR combination (*"[the] [flight]"*, *"[all] [morning flights]"*, *"[just the] [one way flights]"*); the second makes an NP out of a bare plural NBAR, (*"Wednesdays"*, *"travel arrangements"*, *"Delta flights"*). The third rule makes an NP out of a bare NBAR whose `mass` feature has value `y`; thus the grammar and lexicon have to be arranged so that singular NBARS which may legitimately appear bare as NPs are `mass=y`. This is satisfactory for NBARs like *"information"*, or *"breakfast"* which are indeed intuitively mass nouns; it is rather less pleasing that compound NBARs like *"Sunday night"* or *"Pittsburgh airport"* are also deemed to be `mass=y` to make them eligible for application of the rule. In a later version of the grammar it might be desirable to revise the feature system to make analysis of this type of phrase correspond more closely to pre-theoretic intuitions.

We now turn to the six basic NBAR rules,

```
NBAR -> ADJP NBAR

NBAR -> NBAR NBAR

NBAR -> NP:[name=y] NBAR

NBAR -> NBAR CONJ NBAR

NBAR -> NBAR NBAR NBAR

NBAR -> NP NBAR NBAR
```

The first rule licenses prenominal modification of an NBAR by a non-superlative adjective, e.g., *"[direct] [flights]"*, *"[early] [weekday flights]"*, and is unproblematic. The second and third rules, for compound nominals, both contain some infelicities.

The second rule, which allows combination of two NBARS to form a compound NBAR, is the one used to analyze NBARS like *"[afternoon] [flights]"* or *"[limousine] [service]"*. For efficiency reasons, the first daughter NBAR is restricted by feature settings to be lexical; thus the readings of e.g., *"ground transportation information"* and *"early morning flight"* as *"[ground transportation] [information]"* and *"[early*

morning] [flights]" are not licensed by the rule. The first of these is a sufficiently serious problem that a special rule has been added to cover the left-branching reading of a combination of three NBARS (see below), while the other so far appears unimportant.

The third basic NBAR rule permits compound nominals formed by combining an NP and an NBAR. There are two problems involved with the feature settings in this rule. Firstly, the `mass` feature in the mother is uninstantiated, to allow NP+NBAR combinations to count as `mass=y` and appear as NPs, as explained above. Secondly, the daughter NP is constrained to be `name=y`. It is indeed the case that in most instances of application of the rule in the corpus the NP is a name, e.g., *"[Monday] [flights]"*, *"[Boston] [ground transportation]*. However, there are also a fairly large number of cases like *"the [[five forty five] [flight]]"* or *"the [[A P fifty seven] [restriction]]"* which the rule is also intended to cover; for this to be possible, time of day NPs like *"five forty five"* and code NPs like *"A P fifty seven"* consequently need to be `name=y`. This is arguably correct, but sufficiently unintuitive that one feels it may eventually cause problems elsewhere in the grammar.

The fourth rule, which constructs conjoined NBARs (e.g., *"all [flights and fares]"*, is much less frequent than the others and is unproblematic. The fifth and sixth rules deal with left-branching compounds of three elements, and are best regarded as temporary grammar hacks.

The remaining rules all have to do with DETs, which in the CLE grammar include ordinals and superlative adjectives. First, there is a rule for forming superlative adjective phrases, and another for making a superlative ADJ into an ORDINAL: neither of these is problematic.

```
ADJP -> most/least ADJ

ORDINAL -> ADJP:[form=superlative]
```

Finally, we look at the five rules for complex DET expressions:

```
DET -> the ORDINAL

DET -> PREDET DET

DET -> NUMBER

DET -> less/more than NUMBER

DET -> POSSESSIVE
```

The first rule, which is by far the most common of the five, combines an occurrence of *the* with an ORDINAL to form an "ordinal DET", e.g., *"the first"*, *"the longest"*, *"the least expensive"*; the phrase-structure for e.g., *"the cheapest Delta flight"* will consequently be

$$[_{DET}\ the\ cheapest]\ [_{NBAR}\ Delta\ flight]$$

This non-standard analysis is discussed at greater length in Section 9.1.1. The second rule combines a PREDET (e.g., *all*, *only* or *just*) with a DET to form a DET; the daughter DET is in practice always *the*. The third, fourth and fifth rules are straightforward.

9.3.2 Time and date NPs

Time and date expressions are common in the corpus, accounting for about 10% of all non-recursive NPs. The rules are fairly simple, and give nearly complete coverage. By adding a feature to the NUMBER category to distinguish between different types of number expression, it was possible to code the rules (in particular, those for time expressions) fairly compactly: one slightly unobvious trick which we found useful was to allow expressions like *"oh five"* to count as a special type of number.

Before the start of the project, the coverage for date expressions was fairly complete, though a few rules for less frequent date constructions needed to be added. The rules for time of day expressions were written during the course of the project, but are domain-independent, and have already been used in other applications (Lewin *et al.*, 1993).

Examples of the types of time and date expression covered, listed in descending order of frequency of occurrence, are shown below. The rules actually permit more types of expression, though several were not observed to occur in the corpus:

"eight o'clock"

"September fifteenth"

"seven fifty five a m"

"three"

"five forty"

"next Wednesday"

"Tuesday October first"

"July eighteen"

"July twenty fifth nineteen ninety one"

"Monday November eleventh nineteen ninety one"

"July the fourth"

"the tenth of November"

"sixteen hundred hours"

"fifteen July"

"Wednesday the twenty first"

9.3.3 "Code" NPs

NPs involving spelt-out alphanumerical codes are also common in the corpus, and make up about another 10% of the non-recursive NPs. The grammar deals with code NPs as follows. There are two domain-independent rules for code NPs, and one domain-independent rule for CODE,

```
NP -> CODE
```

```
NP -> NBAR CODE

CODE -> NUMBER
```

These are supplemented by a set of seven domain-specific rules for CODE, which define the sequences of letters, numbers and names commonly occurring as codes in the corpus. The rules permit codes to be any of the following: sequences of one, two or three spelt-out letters; sequences of one or two spelt-out letters or a name, followed by a number; or any of the above preceded by the word *number*. For the purposes of these rules, the word *slash* is treated as a letter. Examples of types of phrase covered by these rules, bracketed to show phrase-structure, follow.

$[_{NP}\ [_{NBAR}\ American\ flight]\ [_{CODE}\ four\ eight\ seven]]$
$[_{NP}\ [_{CODE}\ D\ L\ seven\ four\ six]]$
$[_{NP}\ [_{NBAR}\ flight]\ [_{CODE}\ [_{NAME}\ Delta]\ one\ thirty]]$
$[_{NP}\ [_{NBAR}\ flight]\ [_{CODE}\ four\ seven\ six]]$
$[_{NP}\ [_{CODE}\ E\ A]]$
$[_{NP}\ [_{NBAR}\ fare\ code]\ [_{CODE}\ F\ N]]$
$[_{NP}\ [_{NBAR}\ restriction]\ [_{CODE}\ A\ P\ slash\ five\ five]]$
$[_{NP}\ [_{NBAR}\ flight]\ [_{CODE}\ number\ seventeen\ sixty\ five]]$
$[_{NP}\ [_{NBAR}\ airline]\ [_{CODE}\ U\ S]]$
$[_{NP}\ [_{NBAR}\ Delta\ flight]\ [_{CODE}\ number\ seven\ oh\ nine]]$
$[_{NP}\ [_{CODE}\ two\ oh\ four\ five]]$

9.3.4 Bare determiner NPs

About 2% of all non-recursive NPs are bare determiners, formed using the rule

```
NP -> DET
```

A feature limits the range of DETs that can appear on the right-hand side. The DET is most commonly lexical, e.g., *"any"*, *"which"*, *"one"*, *"both"*, *"either"*, *"the same"*, *"some"*, *"how many"* or *"all"*. It can also be an "ordinal determiner", e.g., *"the cheapest"*, *"the latest"* or *"the least expensive"*.

9.3.5 "Kind of" NPs

NPs formed using the "kind/sort/type of" construction are also common, and account for another 2% of the non-recursive NPs; typical examples are *"what kind of plane"*, *"what sort of ground transportation"* or *"the type of aircraft"*.

The words in question all have lexical entries subcategorizing for *of* and an NBAR, and the construction can thus be analyzed using the general rule

```
NBAR:[subcat=[]] -> NBAR:[subcat=[COMP]]
                    COMP
```

which combines an NBAR with its complements to form a larger NBAR with an empty subcat list.

9.3.6 Special non-recursive NP constructions

About 2% of the non-recursive NPs are analyzed using idiosyncratic rules which should arguably not be part of the general grammar. Only three such rules are used:

```
NP:[baresing=y] -> NBAR:[num=sing]

DET -> ORDINAL

NP -> ADJP NP:[name=y]
```

The first constructs an NP out of a bare singular NBAR, e.g., *"fare"* or *"cost"*; NPs of this type are often acceptable in colloquial spoken English. To limit the number of extra analyses created by the rule, the NP it produces is marked with a special feature, which is only accepted by a small number of other rules (see Section 9.8.2). In particular, bare singular NPs can occur alone as elliptical NP phrases.

The second rule, whose motivation is similar, constructs a DET out of a bare ORDINAL, thus permitting NPs like *"latest flight"* or *"greatest fare"*. These expressions are again dubious as standard English, but occur with reasonable frequency in the corpus.

The third rule permits pre-modification of a name by an ADJP to form an NP; it is only used for phrases of the form *"downtown X"*, where X is a city-name. For the purposes of this rule, *downtown* is consequently given an entry as an ADJP.

9.4 Recursive NPs

The structure of the recursive NPs in the domain is on the whole fairly simple. As with the non-recursive NPs, we begin by breaking them down into types, giving examples of each type together with its approximate frequency of occurrence. About 1% of the recursive NP constructions occurring in the corpus are not covered by the grammar.

Basic recursive NPs About 87% of the recursive NPs consist of a non-recursive NP which has been post-modified by a PP, a VP, a relative clause, an ADJP, or some combination of these. Examples: *"the flights from Dallas to Boston"*, *"United flights between Boston and Denver departing at nine a m"*, *"a flight that leaves Atlanta before noon"*, *"the fares available for Eastern flight two oh five"*.

Conjoined NPs About 10% of the recursive NPs are formed using a conjunction rule. The majority arise in connection with the "between A and B" construction.

Sentential NPs About 2% of the recursive NPs are sentential NPs, in practice either embedded questions or *to*-infinitive VPs. Examples: *"to fly from Boston to Baltimore on a Saturday"*, *"to get from Denver to Oakland"*, *"what city they stop in"*, *"if Delta flight two ninety six serves breakfast"*.

"Quote apposition" NPs About 1% of the recursive NPs are constructions of the type we dub, for want of a standard term, "quote appositions": a typical example would be *"the designation Y N"* or *"the fare code F"*. The relatively common occurrence of the construction is clearly due to the frequent occurrence of code expressions in the domain.

9.4.1 Basic recursive NPs

About 87% of the recursive NPs in the corpus are constructed from a non-recursive NP postmodified by a PP, a relative clause, a VP, an ADJP, or some combination of these. The following six rules and one schema are used to capture these constructions:

```
NP -> NP PP

NP -> NP REL

NP -> NP VP:[vform=(ing\/passive)]

NP -> NP ADJP

REL -> S:[type=rel,gapsin=[],gapsout=[]]

REL -> S:[type=normal,gapsin=[np:_],gapsout=[]]

ADJP -> ADJ:[subcat=COMPS]
        COMPS
```

The first two rules are unproblematic. Features are used on the third rule to restrict the VP to ones whose main verb is either a progressive or a passive; the second case covers reduced relatives like *"[a meal] [served on a plane]"* or *"[the type of aircraft] [used]"*. The fourth rule uses a feature to restrict the ADJP to those which are marked as not headfinal. This works well for expressions like *"[the fares] [available for Eastern flight two oh five]"*, where *available for Eastern flight two oh five* can very reasonably be regarded as a non-headfinal ADJP. It is somewhat less clear that it correctly treats expressions like *"[all flights between Boston and San Francisco] [nonstop]"*, since this involves treating *nonstop* as potentially non-headfinal. In order to accommodate examples like this one, *nonstop* and a few other similar ADJPs (*first class*, *one way* and *round trip*) have the headfinal feature unset.

The fifth and sixth rules cover relative clauses. The fifth is for relatives introduced by a relative pronoun, e.g., *"flights [that serve breakfast]"*, *"flights [where the round trip fare is under one thousand dollars]"*. Setting the feature `type` to `rel` ensures that the S is introduced by an element that can serve as a relative pronoun (see Section 9.8.1). The sixth rule is for relatives without a relative pronoun, e.g., *"the cheapest fare [I can get] from Dallas to Denver"*. The final rule schema is for adjectives which take a complement: the only common case in the corpus is *available*, which is treated as subcategorizing for a PP.

9.4.2 Conjoined NPs

About 10% of all recursive NPs use a conjunction construction. The following rules are used, of which only the first is at all common:

```
NP -> NP CONJ NP

NP -> NP NP
```

The first rule handles normal conjunction of two NPs, e.g., *Boston and Atlanta*. The second rule is used in combination with the first to analyze constructions like *Boston Atlanta and Denver*. It creates a conjoined NP, marked with a feature which only allows it to be used to appear as the leftmost daughter in an application of the first rule.

9.4.3 Sentential NPs

About 2% of all recursive NPs are sentential; the grammar allows *to*-infinitive VPs like *"to get from Atlanta airport into the city of Atlanta"*, and embedded questions like *"what city they stop in"* and *"if these flights serve meals"*. Three rules are used,

```
S:[type=norm] -> VP:[vform=inf]

NP -> S:[type=q,inv=n,whmoved=_]

NP -> COMPLEMENTISER S:[type=norm]
```

The first rule makes a *to*-infinitive VPs into an S, so that it can then be used by the other rules. The second rule covers sentential NPs which are either *to*-infinitive VPs or embedded WH-questions; the third rule is for embedded Y-N questions. The features used are explained in Section 9.8.1.

9.4.4 "Quote apposition" NPs

About 1% of all recursive NPs are "quote appositions", NPs consisting of a non-recursive definite NP followed by a code expression. Examples are *"the fare code Q W"* or *"the abbreviation U S"*. These expressions differ both syntactically and semantically from the "code expressions" described earlier. For example, *"flight U A one oh four"* means "the flight whose code is U A one oh four", while *"the code U A one oh four"* means "the code which is equal to U A one oh four". There is a simple rule to deal with these, of the form

```
NP -> NP CODE
```

9.5 Preposition phrases

The structure of preposition phrases in the corpus is also fairly simple. It is difficult to estimate the degree of coverage of PP constructions, because of the substantial grey

area where an NP appears to be functioning as a preposition-less PP; depending on the degree to which one regards these as acceptable, coverage is between about 98% and 99.7%. "Preposition ellipsis" is discussed further below, in Section 9.9.

As explained in Section 9.1.1, PP modification of NPs and VPs makes use of the "big PP" rule

```
PP -> PP PP
```

This rule can be applied recursively, to make any sequence of PPs into a "big PP". There are five types of PP, all of which can either occur on their own or in big PPs:

Normal PPs Normal PPs, consisting of a preposition followed by an NP, make up 93% of all single PPs. For example: *"from Dallas"*, *"to downtown Pittsburgh"*, *"on a flight from Boston to Denver"*.

Temporal NPs "Temporal" NPs (i.e., NPs whose `temporal` feature has the value `y`) can also act as PPs, and make up about 5% of all PPs. Examples: *"what time"*, *"that afternoon"*, *"Thursday"*, *"July twenty third"*.

"A to B" PPs Expressions of the form "A to B", with A and B names, account for a further 1.5% of all PPs, e.g., *"Baltimore to Philadelphia"*, *"Atlanta to Pittsburgh"*.

Stranded prepositions About 0.5% of all PPs are stranded prepositions, i.e., PPs from which the NP has been removed by a movement rule, leaving a lone preposition.

Conjoined PPs There are a small number of examples of conjoined PPs, e.g., *"in Boston and in Baltimore"*, *"after five p m and before eight a m"*.

The above constructions are handled by the following rules, none of which are problematic:

```
PP -> P NP

PP -> NP:[temporal=y]

PP -> NP:[name=y] to NP:[name=y]

PP -> PP CONJ PP
```

9.6 Numbers

Numbers occur frequently in the corpus: they can be cardinals (*"[one thousand one hundred] dollars"*), ordinals (*"September [twenty first]"*) or codes (*"D L [seven six four]"*). The rules for numbers are simple but fairly numerous; they seem to give a coverage of the domain which is virtually complete, the only apparent holes being a few dubious expressions for codes (see Section 9.9).

Examples of some types of numbers covered appear below:

"five eight nine"

"four fifty nine"

"twenty first"

"fifty five"

"ten twenty eight"

"one seven nine three"

"eight thirteen"

"four hundred"

"one thousand"

"three three"

"fourteen nineteen"

"thirty seven forty nine"

"three hundred and thirty six"

9.7 Verb phrases

The rules for verb-phrases are certainly the most complex part of the grammar, mirroring the wide range of variety found in verb-phrase constructions; the proportion of VPs in the corpus which are outside grammatical coverage, at about 1%, is not, however, higher than for most constituents.

We will divide up the material along three dimensions: firstly by the type of main verb (transitive, intransitive, modal, etc.); secondly by the types of modification and transformation of the verb phrase (WH-moved, passivized, PP modified, etc.); and finally by context of occurrence of the verb phrase (predicate in clause, imperative, post-nominal modifier, etc).

Perhaps the main justification for adopting a principled linguistic approach to grammar coverage is that different possibilities along each of these three dimensions can combine freely, giving rise to the large observed range of surface constructions.

9.7.1 Types of verb

Classifying first by main verb, 13 types of verb occur with reasonable frequency:

"Be" as main verb This is the most common type of verb in the corpus, accounting for about 23% of all verb occurrences. The forms *'s*, *am*, *are*, *be* and *is* all occur.

Transitives Transitive verbs are about equally common and make up another 23% of all verb occurrences. Common examples are *have*, *like* and *leave*.

Ditransitives About 15% of all verb occurrences; the only commonly occurring ones are *show*, *tell*, *give* and *find*.

Intransitives About 14% of all verb occurrences: common examples are *fly*, *arrive*, *leave* and *go*.

Modal auxiliaries About 8% of all verb occurrences. The commonly occurring modals in the corpus are *would*, *could*, *can*, *may* and *will*.

Auxiliary "do" and "does" About 7% of all verb occurrences.

Verbs taking to-infinitives About 5% of all verb occurrences. The commonly occurring examples are *like*, *need* and *want*.

Auxiliary "be" About 1.9% of all verb occurrences. The forms *am*, *are*, *be* and *is* occur, with both passive and *-ing* VP complements.

Particle verbs About 0.5% of all verb occurrences. The most common are *stand for*, *get off* and *stop over*.

Verbs taking extraposed sentential complements About 0.5% of all verb occurrences. The only examples are *cost* and *take*, e.g., *"How much does it cost to fly on American from Dallas to Baltimore"*.

Ditransitive verb with embedded question About 0.25% of all verb occurrences are examples of *show* or *tell*, used as in *"Can you tell me what city they stop in?"* or *"Can you show me what's available?"*.

Ditransitive verb taking PP complement About 0.2% of all verb occurrences are examples of *tell about*, as in *"tell me about limousine services"*.

Transitive verb with embedded question There are a small number of occurrences of *know* used as a question-embedding verb, as in *"I need to know what flights leave Atlanta on Sunday evening and arrive in Baltimore"*.

The "basic" verb phrase consists of the main verb together with its complements (if any) and is formed, irrespective of verb-type, using the central rule-schema also discussed in Section 9.3 of Agnäs *al* (1991):

```
VP -> V:[subcat=COMPS]
      COMPS
```

The only type of verb appearing in the corpus whose complement requires special grammar rules is *be* as a main verb, for which five rules exist; *be* subcategorizes for a constituent called a COMP, which has dubious intuitive validity and is probably best regarded as a grammar-writer's fiction.

The COMP rules, which follow, use a feature which has access to the subject of the clause in which *be* is the main verb, and can be used to distinguish *be* with a dummy *there* subject (existential *be*) from *be* with a normal subject (predicative *be*).

The rules are

```
COMP:[subjform=normal] -> NP

COMP:[subjform=normal] -> ADJP

COMP:[subjform=normal] -> PP

COMP:[subjform=there] -> NP

COMP:[subjform=there,gapsin=[np:_],gapsout=[]] -> PP
```

The first three rules allow predicative *be* to take a complement which can be an NP, (*"what are [the fares for flight four fifty nine]"*), an ADJP (*"which of these flights are [nonstop]"*), or a PP (*"is one of the flights [on a seven four seven]"*). The fourth rule is the most common case for existential *be*, in which the complement is a simple NP, e.g., *"is there [any ground transport]"*. The fifth rule is a hack which handles the fairly common construction exemplified in *"what flights are there from Atlanta to Baltimore"*. Here, it seems reasonably clear that *from Atlanta to Baltimore* should attach semantically to *what flights*: the rule achieves this end by pulling the gap associated with *what flights* off the gaps list, and combining it with the PP. A more principled solution, which may be introduced in the future, would be to allow the daughter NP in the PP post-modification rule to be a gap.

9.7.2 Transformations and modifications of verb phrases

We next consider the different ways in which the basic verb phrase can be transformed or modified.

PP post-modifier About 15% of all VPs have a PP modifier following the basic verb phase, e.g., *"go from Denver to Atlanta"*, *"is used on U A five five one"*, *"leaving Atlanta in the afternoon"*.

Movement About 12% of all VPs have their word-order changed by some kind of movement phenomenon, normally occurring in the context of a question or relative clause. Typical examples are the VPs (bracketed) in *"what cities [does Continental service]"*, *"the earliest flight you [have on Wednesday September fourth]"*.

Modal or higher verb About 8% of all VPs contain a modal or higher verb, e.g., *"would like to go at ten a m"*, *"want to see the cheapest flight from Denver to Pittsburgh"*, *"will be served on U A ninety one"*.

ADVP pre-modifier About 5% of all VPs have an ADVP preceding the basic VP. This is most commonly *please* or *also*, e.g., *"please repeat your answer"*, *"also give me a list of flights between Oakland and Boston"*.

Passivization About 2.5% of all VPs are passivized, e.g., *"is served on flight one six nine"*, *"offered in this flight"*, *"is being used on flight number ninety eight"*.

Conjoined VP About 1% of all VPs involve a conjunction, e.g., *"departing San Francisco and arriving in Oakland"*, *"want to leave Philadelphia and arrive in Atlanta on a Thursday"*.

ADVP post-modifier About 1% of all VPs have an ADVP modifier following the basic VP, e.g., *"would like to see the flights from Baltimore to Philadelphia again"*, *"want to fly from Baltimore to Dallas round trip"*.

-ing VP post-modifier About 0.5% of all VPs have an *-ing* VP used as a modifier after the basic VP, e.g., *"go from Denver to Dallas leaving after three p m"*, *"depart from San Francisco heading towards Boston after noon"*.

to VP post-modifier A small number of VPs have a *to* VP used as an adverbial post-modifier, e.g., *"leaving on Tuesdays from Denver [to go to Boston]"*.

Negation A small number of VPs are negated, e.g., *"do not go through Oakland"*.

The structured nature of the grammar allows all these possibilities to be captured with the following small set of rules:

```
VP -> VP PP

VP -> VP ADVP

VP -> ADVP VP

ADVP -> VP:[vform=ing]

ADVP -> VP:[vform=to]

VP -> VP CONJ VP

VP -> not VP

NP:[gapsin=[np:_|Rest],gapsout=Rest] -> []

PP:[gapsin=[pp:_|Rest],gapsout=Rest] -> []

ADJP:[gapsin=[adjp:_|Rest],gapsout=Rest] -> []

ADVP:[gapsin=[advp:_|Rest],gapsout=Rest] -> []
```

The first seven rules are all straightforward. The first three permit VPs to be post-modified by PPs and ADVPs and pre-modified by ADVPs; the fourth and fifth make *-ing* and *to* VPs into ADVPs, thus allowing them to act as VP modifiers via the second and third ones. The sixth and seventh allow VP conjunction and VP negation. The last four rules handle *both* WH-movement of NPs, PPs, ADJPs and ADVPs, *and* passivization of VPs (see the last sub-section of Section 9.1.1). Note also that modal verbs

require no special rules: they are viewed as subcategorizing for VP complements, and are analyzed by the basic VP rule which constructs a VP from a verb and its complements.

9.7.3 Verb phrase contexts

Finally, we consider the possible contexts in which a VP can occur. Apart from combining with an NP to form a clause (the most common case), the following possibilities exist:

Imperatives About 30% of all VPs occur free as imperatives, e.g., *"Show me the flights from Dallas to Boston"*.

-ing VP modifiers About 6% of all VPs are *-ing* VP modifiers to NPs, e.g., *"flights going to San Francisco"*.

Reduced relatives About 0.6% of all VPs are reduced relatives, e.g., *"a meal served on a plane"*.

to-VP as NP About 0.4% of all VPs are *to*-infinitive VPs functioning as NPs, e.g., *to travel from Oakland airport to downtown*.

The rules needed to realize these possibilities are described in other sections.

9.8 Clauses and top-level utterances

Having discussed the other constituents, we finally consider the structure of clauses and top-level utterances. (We will use the terms "utterance" and "top-level utterance" interchangeably to designate the start-symbol of the grammar). The following types of utterance exist:

Unmoved WH-questions WH-questions not exhibiting movement are the most common type of utterance in the corpus, accounting for about 29% of the examples. E.g. *"what is the cheapest flight from Boston to San Francisco"*, *"what flights go from Pittsburgh to Baltimore after eight o'clock next Wednesday"*, *"what type of aircraft is used on the five forty flight from Philadelphia to Dallas"*.

Imperatives Imperatives are about equally common, accounting for 28% of the utterances. For example, *"list all flights on United from San Francisco to Boston"*, *"show me ground transportation in Dallas please"*, *"now show me the flights from Denver to Philadelphia on a Saturday"*.

Y-N questions About 12% of the utterances are Y-N questions, e.g., *"is U S Air flight four seventy six a nonstop flight"*, *"all right do you have a flight from Atlanta to Boston"*, *"okay can you tell me the flight cost between Denver and Atlanta"*, *"could you repeat that please"*.

Declarative sentences About 9% of the utterances are declarative sentences, though naturally most of these have the force of requests, commands or questions. Typical examples are *"I need to know what flights leave Atlanta on Sunday evening and arrive in Baltimore"*, *"I need information on a flight from Boston to Denver"*, *"yes I want to go to San Francisco late that afternoon"*.

Moved WH-questions WH-questions with movement are much less frequent than ones without, accounting for only about 7% of the utterances. Despite this, moved WH-questions exhibit considerably more variety than unmoved ones, e.g., *"what cities does Continental service"*, *"what ground transportation is there from Denver"*, *"how much does it cost to fly on American from Dallas to Baltimore"*, *"how do I get from Pittsburgh airport to downtown Pittsburgh"*, *"how long is the flight from Oakland to Washington D C"*, *"what time are the flights leaving from Denver to Pittsburgh on July seventh"*.

Elliptical NPs Elliptical utterances are common: more than half of them are NPs, possibly introduced by an interjection or a phrase like *how about* or *what about*, or followed by *please*. These account for another 7% of the corpus. Typical examples are *"the most expensive flight between Boston and Philadelphia"*, *"six fifty three a m"*, *"flights from Atlanta please"*, *"how about twelve thirty p m"*, *"what about a flight from Boston to San Francisco stopping in Denver"*.

Elliptical PPs and ADVPs A further 6% of the corpus consists of elliptical PPs and ADVPs, with PPs accounting for abut two-thirds of these. As with elliptical NPs, they can be accompanied by a interjections and phrases like *please* or *how about*. Examples: *"from Pittsburgh"*, *"how about on Sunday night"*, *"and for Lufthansa"*, *"one way"*, *"round trip please"*.

Compound utterances The final 2% of the corpus consists of utterances which at least arguably can be regarded as sequences of two or more sub-utterances, linked by relations which are defined pragmatically by the context. The grammar has as yet only a fairly rudimentary coverage of these constructions. Typical examples follow, with brackets indicating the proposed boundaries between the sub-utterances: NPs, PPs, declarative clauses, WH-questions, Y-N questions, and imperatives.

$[_{PP}\ from\ Denver\ to\ Pittsburgh\ on\ April\ twenty\ first]\ [_{dcl}\ I\ need\ the\ cheapest\ flight]$

$[_{NP}\ flight\ E\ A\ two\ ten]\ [_{imp}\ give\ me\ information\ on\ the\ price]$

$[_{PP}\ of\ those\ flights]\ [_{imp}\ show\ me\ the\ flights\ serving\ breakfast]$

$[_{dcl}\ I\ wish\ to\ fly\ from\ Boston\ to\ Washington]\ [_{imp}\ please\ find\ an\ airline\ for\ me]$

$[_{dcl}\ I\ am\ planning\ a\ trip\ to\ Pittsburgh\ and\ I\ live\ in\ Denver]\ [_{whq}\ can\ you\ help\ me]$

$[_{dcl}\ some\ of\ these\ flights\ have\ stops]\ [_{ynq}\ can\ you\ tell\ me\ what\ city\ they\ stop\ in]$

9.8.1 Clauses

It is clear from the percentage-figures shown in the text above that clauses (which include WH-questions, Y-N questions, imperatives and declarative sentences) are the most common type of utterance, and we will consequently consider first the rules used for forming them. We begin with the five rules which do not have to do with WH-movement or subject-verb inversion:

```
S:[type=T] -> NP:[type=T] VP

S:[type=imp] -> VP:[vform=inf]

S -> ADVP:[advtype=sentential] S

S -> S ADVP:[advtype=sentential]

S -> S CONJ S
```

The `type` feature on S can have values `q` (question), `r` (relative) or `norm` (normal, the default). `q` or `r` values are passed up from the subject, when the clause does not exhibit movement, and otherwise from the fronted element as explained below.

The first of the rules just presented is the basic one which combines subject NP and VP to form a clause; the second forms an imperative clause from a VP with infinitive main verb. The third and fourth allow pre- and post-modification of clauses by sentential adverbs, and the fifth forms conjoined clauses. All of these are straightforward. We now consider the six rules used for dealing with inversion and WH-movement, which are more complex in structure:

```
S:[inv=y] -> V:[subcat=List] NP VP:[sai=movedv:[subcat=List])]

V:[subcat=[vp:VP],sai=movedv:[subcat=[vp:VP]]] -> []

V:[subcat=[comp:COMP],sai=movedv:[subcat=[comp:COMP]]] -> []

S:[whmoved=y,type=T,inv=I] ->
   NP:[type=T] S:[inv=I,gapsin=[np:_],gapsout=[]]

S:[whmoved=y,type=T,inv=I] ->
   PP:[type=T] S:[inv=I,gapsin=[pp:_],gapsout=[]]

S:[whmoved=y,type=q,inv=I] ->
   ADJP:[wh=y] S:[inv=I,gapsin=[adjp:_],gapsout=[]]
```

The first three rules cover subject-auxiliary inversion; two new features play important roles. The `inv` feature distinguishes clauses with inverted word-order from normal ones; the `sai` (subject-auxiliary inversion) feature is used to pass information about the fronted verb to the place in the VP where it would have occurred in an uninverted clause.

The first rule defines the basic structure of a clause with subject-auxiliary inversion, as a sequence of auxiliary verb, subject, and inverted verb phrase. The `sai` feature propagates the fronted verb down through the verb phrase to reach the main verb: there, it can be picked up by either the second or third rule.

The second covers all the verbs which can be fronted except copula or predicate *be*; all other such verbs subcategorize for a VP. The third rule handles *be* (see also the discussion of COMP rules in Section 9.7.1).

The fourth, fifth, and sixth rules cover movement and make essential use of the `whmoved` feature. The fourth and fifth are for fronted NPs and PPs. These rules can be used for WH-questions occurring as main clauses, embedded WH-questions, or relatives, the type of clause being determined by the `inv`, `type` and `whmoved` features. Thus for example embedded WH-questions are `whmoved=y`, `inv=n` and `type=q`, while relative clauses are `whmoved=y`, `inv=y` and `type=r`. The `type` feature is passed up from the fronted constituent; so for example *that* has an NP entry with `type=r`, allowing it to introduce relative clauses but not inverted WH-questions. In contrast, the NP entry for *what* has `type=q`, giving it the opposite distribution.

The sixth rule is for fronted ADJPs: these can only introduce questions, e.g., *"How expensive is the San Francisco limousine service"*.

9.8.2 Utterances

A top-level utterance can either be some type of clause or elliptical phrase, or a sequences of two or more such items. We consequently distinguish three types of constituent: elliptical phrases, utterance units (a term we will use to subsume both single clauses and single elliptical phrases), and utterances.

We consider the rules for each type of constituent in turn, starting with those for elliptical phrases:

```
PHRASE -> PP

PHRASE -> ADVP

PHRASE -> NP:[baresing=_]

PHRASE -> NP:[baresing=y] NP:[name=y]
```

The first three rules allow PPs, ADVPs and NPs to stand alone as elliptical phrases. The first two rules are straightforward; the third has the `baresing` feature (see Section 9.3.6) uninstantiated, to allow bare singular NPs to be elliptical NP phrases.

The fourth rule covers the spoken-language construction, common in the ATIS corpus, typified by utterances like *"cost D L eight five two"* or *"ground transportation Atlanta"*. In expressions like these, the intended semantics places an elided vague preposition between the two NPs: thus the interpretation of the example utterances will be approximately *"cost D L **for** eight five two"* and *"ground transportation **for** Atlanta"*.

The next group of rules cover "units", which represent the smallest possible free-standing utterance. Most utterances in fact consist of just one unit. The first three rules are for phrasal units:

```
UNIT -> PHRASE

UNIT -> CONJ PHRASE

UNIT -> how/what about PHRASE
```

The first rule allows a UNIT to be a single phrase; the second is for phrases introduced by a conjunction (e.g., *"and from Dallas to Atlanta"*, *"and United"*); and the third for phrases introduced by *what about* or *how about*, e.g., *"how about for Eastern Airlines"*, *"what about a seven three four"*.

The five remaining rules are for clausal units:

```
UNIT -> S:[type=imp]

UNIT -> S:[type=q,inv=y,whmoved=y]

UNIT -> S:[type=q,inv=n,whmoved=n]

UNIT -> S:[type=norm,inv=y,whmoved=n]

UNIT -> S:[type=norm,inv=n,whmoved=n]
```

These cover, in order, imperative clauses, WH-word questions with WH-movement, WH-questions without WH-movement, Y-N questions, and simple declarative clauses. The features used are explained in Section 9.8.1.

The last group of rules are for utterances:

```
UTTERANCE -> UNIT

UTTERANCE -> INTERJECTION UNIT

UTTERANCE -> PP UNIT

UTTERANCE -> UNIT INTERJECTION

UTTERANCE -> CONJ UNIT

UTTERANCE -> NP UNIT

UTTERANCE -> S UNIT
```

The first rule is for utterances consisting of a single unit; the others cover the most commonly occurring cases in which an utterance is built up from a unit and some other constituent. These rules have an *ad hoc* character, and are the only ones in the grammar that cannot reasonably be claimed to be linguistically motivated.

9.9 Coverage failures

In the previous sections, we have described the constructions currently covered by the grammar. We will now consider the corpus utterances which are still outside coverage. Our main goal will be to show that the majority of these could be covered by fairly simple and systematic extensions to the present framework; this includes many examples of "telegraphic" and other constructions which, though typical of spoken English, would not be considered strictly grammatical by most native speakers.

Confining ourselves to the development corpus, which consists of the 4615 A or D class ATIS sentences of 15 words or less available at the start of the project (minus 600 reserved for testing), the grammar currently covers all but 426 (9.2%). Of these, 93 (2.0%) are currently classed as malformed; the other 333 (7.2%) are arguably such that they should be covered by the grammar. The boundary between malformed and well-formed sentences is a vague one, and disputable borderline cases exist. We will start by examining the putatively "grammatical" coverage failures, and then look at the "ungrammatical" ones. Note that some sentences display more than one problem.

9.9.1 "Grammatical" coverage failures

We begin by giving a top-level classification of common coverage failures, with examples; we then discuss each class individually.

Spoken language constructions The most common single type of construction responsible for coverage failure is "telegraphic" spoken language, characterized by omitted articles and/or prepositions, compound utterances formed of a sequence of two or more sentences or elliptical fragments, or some combination. These account for 112 sentences (2.4%); the regularity and ubiquity of these constructions makes it difficult to dismiss them as "ungrammatical", and many are already covered by rules in the existing grammar. Typical examples are *"Show me flights with round trip fare less than one thousand dollars"* (omitted article before *fare*), *"arriving the earliest flight"* (omitted preposition before *the*), or *"all right the nine thirty flight does that have a meal"* (sequence of interjection, NP and question).

Missing lexical entries Missing lexical entries account for 67 sentences (1.5%). Many of these are multi-word phrases, e.g., *by way of* (*"I would like to fly from Denver to Atlanta by way of Pittsburgh"* or *one hundred* (as an aircraft type), e.g., *how many seats in a one hundred*.

"Normal" non-recursive NPs Missing grammatical coverage for "normal" non-recursive NP constructions (i.e., constructions not specific to spoken language) accounts for 30 sentences (0.65%). Most of these are "code" expressions, e.g., *"what restrictions apply to [the Eastern flight eight twenty five]"*, or complex determiners *"Which airline has [the most arrivals in Atlanta]"*.

Conjoined phrases Missing rules for types of conjoined phrase account for 29 sentences (0.65%), e.g., *"List the [arrival and departure] times for these flights"*, *"Show me flights [two oh two and two oh eight]"*.

Modifier expressions Missing rules for modifier expressions (PPs, ADVPs, and *-ing* VPs) account for 26 sentences (0.55%), e.g., *"Are there any flights that arrive [just after five p m]"*, *"Philadelphia to Dallas arriving before one in the afternoon"*.

Word-order problems 24 sentences (0.5%) fail due to lack of coverage of various word-order problems. Typical examples are *"What evening flights do you have available from Baltimore to Philadelphia"*, *"List all flights please from Washington to San Francisco"*.

Other problems 52 sentences (1.1%) fail for other reasons.

We now consider each class in more detail.

Quasi-grammatical spoken language constructions

Later on, we will consider examples of spoken-language constructions which overtly break the accepted rules of English grammar (e.g., agreement between subject and verb). Utterances of this type, however, are considerably less frequent than those containing constructions of unclear grammatical status. In particular, two types of construction stand out. Firstly, it is in spoken English quite common to omit determiners and prepositions when they are clear from context. This results in what is sometimes referred to as "telegraphese", though we will prefer the more descriptive terms "determiner ellipsis" and "preposition ellipsis". The other common type of quasi-grammatical spoken language construction is the multiple utterance, in which several sentences and/or elliptical fragments are juxtaposed in a single utterance, the connections being again clear from context.

The grammar has a limited ability to deal with both types of construction, rules having been added for the simplest and most common cases (see Section 9.8.2). Regarding "telegraphese", an elliptical NP fragment may be a bare singular NP (determiner ellipsis), e.g., *"Cost of a round trip ticket on flight D L one zero five nine"*; an utterance may also consist of a bare singular NP followed by an NP (determiner and preposition ellipsis) e.g., *"Fare D L three one one"*, meaning *"**The** fare **for** D L three one one"*. There are also a number of rules for common types of multiple utterances, e.g., NP+S (*"U S seven seven one Pittsburgh to Philadelphia what is the fare"*) and PP+S (*"On the flights that have stops can you tell me where they stop"*.

Fuller coverage of determiner and preposition ellipsis, which make up over 90% of the sentences intuitively classified as "telegraphese", could be achieved in a principled way by simply adding general "bare singular NP" and "prepositionless PP" rules. The resulting increase in the number of analyses produced would naturally be appreciable, and would as usual result in two problems: first an increase in processing time, and secondly an increased load on the preference component. Inspection of typical sentences suggests however that neither problem is critical. The increase in the number of analyses produced is unlikely to exceed a factor of two or three on most sentences; also, it is noticeable that only a small number of contextually salient words (*cost*, *fare*, *stop*, etc.) tend to occur as head-words in "telegraphic" constructions, which makes it reasonable to hope that collocation-based preference metrics would be able to reject

most spurious extra readings. Our impression is that it would probably be feasible to cover nearly all examples of simple determiner and preposition ellipsis by some variant of this method.

Better coverage of multiple utterances could be achieved by some simple improvements to the existing grammar rules. About 70% of the multiple utterance failures observed in the training corpus fall into one of the following categories:

- Multiple utterance of a type already covered, but with the addition of an interjection or conjunction, e.g., *"[All right] [the nine thirty flight] [does that have a meal]"* (would be covered without interjection *all right*); *"[Thanks] [and] [what's the last flight back from Washington]"* (would be covered with either *thanks* or *and* deleted).

  Modification of the multiple-utterance rules to allow freer insertion of conjunctions and interjections is not difficult.

- Common multiple-utterance patterns currently not covered. There are four specific cases: PP followed by NP, e.g., *"[On July twenty third] [an early flight on American from Philadelphia to San Francisco]"*; S followed by NP, e.g., *"[Is that a nonstop flight] [United Airlines three five five]"*; S followed by S, e.g., *"[I want to fly from Boston to Atlanta] [I would like the cheapest fare please]"*; and two PPs separated by an interjection, e.g., *"[Pittsburgh to Atlanta] [please] [with a stopover in Fort Worth]"*.

  Addition of rules to cover all of these cases would again be completely straightforward.

Missing lexical entries

Lexical entries exist for all items which occur at least three times in the development corpus, and also for most items occurring once or twice. There are two main points of interest regarding the missing lexical entries; firstly, the extent to which they are domain-specific, and secondly, the degree of difficulty involved in adding an appropriate entry. Of the 67 sentences which fail due to missing lexical entries, 47 (70%) fail due to entries which could fairly be described as domain-independent (see Section 9.2). The remaining 20 entries are closely linked to the ATIS domain; of these, 9 are code phrases referring to aircraft types, e.g., *D nine S* or *seven three four*. The other 10 missing domain-specific entries are *a m* as a noun, *boeing* (proper noun), *booking class* (common noun), *cheapest cost* as an adjective, *coach* as an adverb, *economy* as a proper noun, *economic* as a synonym for *economy*, *p m* as an adjective, *round trip* as a noun and *thrifty* as a synonym for *thrift*. None of these would have been difficult to add.

Of the missing domain-independent entries, 11 are function words, 12 are words which would be added to the lexicon as idiosyncratic items with an individual entry, and the remaining 24 are open-category words conforming to implemented lexical paradigms. We consider each class in turn. The missing function words are: *and then* (conjunction), *as possible* (comparative phrase), *aside from* (preposition), *besides* (preposition), *by way of* (preposition), *non* (negating adjectival modifier, e.g., *"non first*

class flights"), *once* (subordinating conjunction approximately equivalent to *when*), *regardless of* (preposition) and *say* (interjection, e.g., *"What flights arrive after say six o'clock"*). All of these could be added easily, with the exception of *non* and *say*.

The missing idiosyncratic entries are: *at night* (PP), *currently* (adverb), *downtown* (adverb, e.g., *"How much does it cost to get downtown from the Atlanta airport by limousine"*), *fine* (interjection), *instead* (adverb), *kind* (noun subcategorizing for NP, e.g., *"What kind of an aircraft is that"*), *kindly* (adverb), *possible* (postmodifying adjective used in conjunction with superlative, e.g., *the earliest flight possible*) and *proper* (postmodifying adjective, e.g., *the city proper*). Of these, the only difficult one is *possible*, which appears to require a special grammar rule.

We finally look at the missing open-class entries, all of which could have been added unproblematically: *add* (ditransitive verb), *far from* (adjective), *L* (name of letter, e.g., *"What does the L stand for under meals"*), *list* (intransitive verb, e.g., *"List from Boston to Atlanta"*, or verb taking embedded question, e.g., *"Please list where these planes stop"*), *look like* (particle verb), *make* (ditransitive verb, e.g., *"Make the date of the flight July second"*), *mention* (transitive verb), *Philadelphia Pennsylvania* (proper noun), *pricing* (common noun), *qualify* (intransitive verb), *quote* (transitive verb), *restricted* (adjective), *San Francisco California* (proper noun), *seating* (common noun), *set* (verb subcategorizing for direct object and *to*-PP, e.g., *"Set the date of the flight to July second"*), *show* (verb subcategorizing for direct object and *about*-PP, e.g., *"Show me about the ground transportation in Boston"*), and *summer* (common noun).

In summary, about 70% of the missing entries were domain-independent, and about 90% could have been added without difficulty.

"Normal" non-recursive NPs

32 sentences failed due to problems with "normal" (i.e., not spoken-language specific) non-recursive NPs. Of these, 14 were due to a number of missing rules for complex determiners of the following forms: *the most/least*, e.g., *"Which airline has the most arrivals in Atlanta"*; possessive followed by a superlative, e.g., *"Show me your earliest flight"*; *the* followed by a number, e.g., *"the four fare classes"*; *the* followed by a number and a superlative, e.g., *"the three earliest flights"* and *at most/least* followed by a number, e.g., *"at least three stops"*. Another 11 sentences failed due to problems with code expressions. It is not clear how many of these are actually well-formed (for example, the sentence *"Is flight number two one seven one four nine first class"* appears to be a restart). The remainder appear to be unusual ways of expressing number codes, e.g., *"flight number one fifteen twenty"*. Finally, two sentences failed due to a simple feature bug in the conjoined NBAR rule, and five sentences failed for assorted other reasons.

Conjoined phrases

29 sentences failed due to missing coverage of conjunction constructions. Of these, 26 could have been solved by addition of simple rules to cover the following types of constituent conjunction: codes, e.g., *"flights [two hundred two and two hundred eight]"*;

PPs, e.g., *"both after twelve p m and before twelve p m"*; complex nominals, e.g., *"the [arrival and departure] times"*; prepositions, e.g., *"into and out of"*; adverbs, e.g., *"one way and round trip"*; and superlative adjectives, e.g., *"the [latest and earliest] flights"*.

There were three genuinely difficult cases in which constituents of different classes were conjoined, e.g., *"Find me a flight [arriving near five p m and nonstop]"*. These could not have been solved without addition of a very large number of low-frequency rules, or major modification of the grammatical framework.

Modifier expressions

26 sentences failed due to missing coverage of modifier expressions. Of these, nine were examples of utterances consisting of a lone *-ing* VP, e.g., *"Arriving in Washington D C"*; five were due to adverbial modification of PPs, e.g., *"just after five p m"*; four were due to due to modifier phrases consisting of a combination of a PP and and an *-ing* VP, e.g., *"From Boston going to Atlanta"*; four were superlative ADVPs of the form *most/least* or *the most/least*, e.g., *"Which airline flies out of Boston (the) most"*; three were superlative ADVPs of the form *the ADVP*, e.g., *"Show me the flight that arrives the earliest"*; and two failed for other reasons. Rules for all the named examples would be easy to add.

Word-order problems

24 sentences fail due to inadequate coverage of word-order phenomena. Seven fail due to problems with movement, typified by sentences like *"What do you have tomorrow morning from Pittsburgh to Atlanta"* or *"What evening flights do you have available from Baltimore to Philadelphia"*. These sentences are interesting, in that they appear to point to a flaw in the current treatment of movement. It would be possible to cover them by allowing the daughter NP in the rules which build an NP from an NP and a post-modifier to be a gap; this involves making a number of alterations in the features present in these rules, so as to limit the number of new readings produced. It is not a trivial change, but one which appears well-motivated, and feasible without drastic reorganization of the grammar.

The remaining 17 sentences exhibiting word-order problems fail due to cases where modifiers occur in positions currently not handled by the grammar. In eight of these, a PP or ADVP occurs immediately after the main verb, e.g., *"Please list for me the flights on United Airlines between Boston and Denver"* (PP) or *"I'm only interested in nonstop flights"* (ADVP). Three more are caused by a bug which currently blocks post-VP modification of WH-questions by sentential adverbs. The modifications needed to cover these cases are easy; of the remaining six sentences, five fail due to presence of an ADVP or PP inside an NP or PP, e.g., *"Please show me the flights again from Pittsburgh to Atlanta"*. This type of construction is, in contrast, not at all simple to accommodate in the grammar.

Other problems

52 sentences fail for other reasons. Of these, ten are caused by lack of rules for nominalizations and conditionals; nominalizations were primarily omitted in the interests of efficiency, and conditionals for no very good reason. Most of the remaining 42 sentences represent problems that could in principle be solved by addition of suitable grammar rules or slight modification of existing ones, but have not been dealt with due to their low frequency of occurrence. Of these 42 sentences, 9 represent problems that occur three times; 16, problems that occur twice; and the remaining 17, problems that occur only once.

9.9.2 "Ungrammatical" coverage failures

92 of the sentences currently outside coverage are fairly clearly ungrammatical in the strong sense of actively breaking established grammatical rules, rather than merely representing constructions not in the grammar.

58 of these sentences are "repairs" not currently handled by the special mechanism used to process this type of disfluency (see Section 4.1 of Agnäs *et al*, 1994). 13 more sentences break subject-verb agreement rules, e.g., *"What are the meal"*); it is possible that the requirement on subject-verb agreement could be relaxed to cover them, but this is a step we have so far been reluctant to take. 10 sentences display forms of ellipsis sufficiently irregular and uncommon that they are difficult to include in a grammar, e.g., *"Me the flights from American"* (ellipsis of main verb) or *"Interested in a flight from Washington to Fort Worth"* (ellipsis of both subject and main verb). Finally, 11 sentences display irregular patterns hard to classify. For example, in the sentence *"What does the origination of flight D L eight four two", is* has presumably been substituted by *does*, but this cannot sensibly be captured in a grammar.

9.9.3 Summary of coverage failures

The main motivation for the detailed presentation given here has been to support our contention that most of the present coverage holes could be filled in a reasonably principled way without large changes to the grammar. The two most frequent single problems, repairs and spoken-language specific constructions, actually appear quite tractable; these between them account for 170 of the 426 coverage failures. Many sentences which currently fail due to missing grammar or lexicon coverage are also fairly regular, in the sense of displaying well-defined patterns that occur several times in the corpus, and could be solved by simple additions or modifications to the grammar. The remaining sentences can be difficult for one of three reasons. They can be sufficiently irregular that they are stamped as "malformed" and ignored; they can contain problems apparently insoluble without major restructuring of the grammatical framework; or they can belong to the "tail" of low-frequency rules, many of which would remain uncovered even if a substantial effort were spent on improving the grammar.

The first and second groups are small. Not counting repairs (which are not really a grammatical problem), only 24 sentences (0.6%) are still classed as malformed. The second group is even smaller: the only clear cases of serious problems with the

grammatical framework are those involving conjunction of dissimilar constituents (3 sentences), and word-order problems where an ADVP or PP modifier to a VP appears inside an NP (5 sentences). The most difficult grammar modification required seems to be the one relating to extraction from NPs described in Section 9.9.1, which is almost certainly a question of a few days of work at most. Combining the counts for the first two groups, the grammatical framework appears strong enough in principle to account for all but about 32 sentences (0.7%) of the development corpus.

The third group of difficult sentences is larger. We will estimate its size by counting all problems that occur less than three times in the development corpus: any known problem that presents no intrinsic difficulties in solution and occurs as frequently as once in every 1500 sentences will surely be fixed eventually given a substantial development effort. Counting the low-frequency examples from the various groups, we have 11 cases of low-frequency multiple utterance patterns; 67 missing lexical entries; 16 problems with non-recursive NP rules; 2 problems with modifier constructions; 1 word-order problem; and 33 "other" problems, making a total of 130 sentences, or 2.8%. In summary, it seems justifiable to claim that the practical coverage limit inherent in the grammar framework used is not worse than 97%, and that coverage at this level could be achieved with only a moderate investment of effort: we are fairly confident that an additional six person-months of labour would be sufficient.

Chapter 10

Swedish Coverage

Manny Rayner, Mats Wirén and Robert Eklund

Note: This chapter is a preliminary draft and will expanded in greater detail for the final version of the report.

10.1 Introduction

Historically, the Swedish grammar and lexicon were adapted from their English counterparts (Alshawi *et al.*, 1991) and they have largely continued to grow in parallel, allowing the large overlap between the structures of the two languages to be exploited. Because of the common origin, and since the English grammar has been described in fair detail elsewhere, we shall concentrate here on the systematic differences between the two grammars rather than attempting to describe the Swedish grammar as an independent entity.

As for the English grammar, the presentation is organized with respect to chunks or levels corresponding to morphology, verbal constructions, clausal constructions, NP constructions, modifier constructions and phrasal constructions.

10.2 Morphology

10.2.1 Declensions/conjugations

In contrast to English, Swedish nouns have more than one declension; Swedish verbs have more than one conjugation.

Unlike English, adjectives inflect by number, gender and definiteness.

Swedish rules:

`v_v_affix` Combine a verb and an affix for the right conjugation

`nbar_nbar_plural` Combine a noun and a plural for the right conjugation

`adj_adj_plural` Make plural form of adjective

`adj_adj_neuter` Make neuter form of adjective

Swedish features:

`synmorphn` Noun declension `synmorpha` Adjective declension `synmorphv` Verb conjugation

10.2.2 Null derivation

In English, the only null derivation needed is that one that derives the imperative/infinitive form from the notional base form. In Swedish, the possibilities are much more numerous and need to be controlled by features.

English rule:

`v_v` Imperative/infinitive from base

Swedish rules:

`v_v_null` General null derivation of verb `nbar_nbar_nullplural` Null plural for 5th declension and some other nouns

Swedish features:

`nullmorphn` Noun has a null plural if set

`nullmorpha` Adjective does not inflect by number and gender if set

`nullmorphv` Verb forms formally identical with base (depends on conjugation)

10.2.3 Umlaut

Swedish has productive vowel mutation with strong verbs and adjectives. (Also on nouns, but here doesn't need to be treated as productive: can be thought of as a straight irregular plural).

Swedish rules:

`adj_adj_omljud` "Comparative stem" for strong adjective. Used for comparative and superlative forms.

`v_v_supine_stem` "Supine stem" for strong verb. Used for supine and past participles.

10.2.4 Adverb from Adjp

Swedish forms adverbs from the singular indefinite neuter form of an adjective

Swedish rule:

`advp_adj_neuter` Adverb from adjective

10.2.5 Verbal Constructions

10.2.6 Lexical passive and deponent verbs

Swedish has lexical passive. Two variants:

- Some verbs are deponent, i.e. formally passive but semantically active.
- Some verbs allow the "impersonal passive" construction using the dummy subject "det".

Swedish rules:

`v_v_passivize` Normal lexical passive

`v_v_Deponent` Formal passive for deponent verb
`v_v_passivize_impersonal` Impersonal passive form of verb (requires dummy subj)
Swedish features:
`deponent` Verb is deponent if set
`impersonal_passive` Verb allows impersonal passive

10.2.7 Separable verbs

Swedish transitive particle verbs form a past participle by attaching the particle before the verb.
Swedish rule:
`v_p_v` Past participle for Swedish transitive particle verb

10.2.8 Lexically reflexive verbs

Swedish, unlike English, has lexically reflexive verbs (verbs which subcategorize for a semantically null reflexive pronoun)
Swedish feature (np):
`reflexive` Marks dummy reflexive pronoun

10.3 Clausal Constructions

10.3.1 Inversion

Swedish, unlike English, allows inversion of any verb.
Swedish requires inversion when a clause is introduced by a sentential adverbial.
Swedish rule:
`s_advp_s_ConjAdvp` Swedish clause introduced by adverb requiring inversion
English rule:
`s_advp_s_SAdvp` English clause introduced by adverb (no inversion)
English features (on v):
`inverts` V can invert
`mainv` V appears as main verb in clause

10.3.2 Mobile adverbs and negation

Swedish "mobile" adverbs (which formally include negation markers) occur after the verb in a main clause, but before the verb in a subordinate clause. "Special" adverbs always appear before the verb. "Conjunctive" adverbs always appear after. If the verb is transitive, it occurs in a main clause, and its direct object is a pronoun, then any mobile adverb occurs after the object.

In view of the above observations, it is natural to consider the verb and adverb (negation) as together forming a constituent. In contrast, it is in English more natural

to consider that the negation marker combines with a VP. This explains the following divergences.

English rule:

`vp_not_vp` Negative verb phrase

Swedish rules:

`v_v_advp_ConjAdvp` Postverbal conjunctive adverb

`v_v_advp_Mobile` Postverbal mobile adverb (main clause)

`v_v_pro_advp_Mobile` Postverbal mobile adverb (main clause) with pronominal object

`v_advp_v_Special` Preverbal special adverb

`v_advp_v_Mobile` Preverbal mobile adverb (subordinate clause)

Swedish features:

`mobile` Adverb feature distinguishing mobile/non-mobile advps

`subordinate` VP/clausal feature distinguishing subordinate clauses

`biff` V feature marking Vs pre-modified by mobile advps

10.3.3 Vad ... för

Swedish allows the vad ... för construction.

Swedish rule:

`np_för_np` "för" PP including gap left by "vad"

10.3.4 Swedish embedded Q with "som"

Swedish requires embedded subject questions to take an extra "som" between the subject and the VP.

Swedish rule:

`s_np_som_vp_EmbeddedQ` Special clause of this type.

Swedish feature (on s):

`somclause` Marks these clauses and only allows them to be used as embedded Qs

10.4 NP Constructions

10.4.1 Definiteness

Swedish, unlike English marks nouns and adjectives for definiteness.

```
rule_counterpart_declared([np,nbar],np_nbar_Def/swe,?).
```

Swedish rules:

`nbar_nbar_def` Definite form of noun

`adj_adj_def_Normal` Definite form of adjective

`adj_adj_def_Sup` Definite superlative form of adjective

Swedish feature (on np, nbar, adjp):

`def` Definite form if set

10.4.2 Bare adjp

Swedish allows adjectives to be used productively as nouns

Swedish rule:

`nbar_adjp_NbarEllipsis` Adjective used as noun

Swedish feature (on nbar):

`adjnbar` Adjectival noun if set

10.4.3 Possessive constructions

English forms the possessive by affixation of a possessive element (’s) to the NP.

Swedish forms the possessive by putting the head N of the NP into the genitive.

English rule:

`possdet_np_poss` NP + ’s is a possessive

Swedish rules:

`nbar_nbar_genitive` Genitive form of common noun

`name_name_genitive` Genitive form of proper noun

`det_np_Genitive` NP with ‘genitive’ feature set is a possessive

`np_det_nbar_code` Special rule for genitive/code combination, e.g. "Deltas flygning tvåhundrafem"

Swedish feature (on nbar and np):

`genitive` Head feature specifying genitive

10.4.4 Compound nominals

English forms nominal compounds with intervening spaces. Hence the compounding rules are grammar rules.

Swedish normally forms nominal compounds without intervening spaces, though in compounds of the form proper noun + common noun the space is optional. Hence compounding rules, with this one exception, are morphology rules.

If the first element of the compound is a noun, then it may appear in a modified form. Possibilities include affixation of "s" (tidszon), hyphen (tur och retur-flygning), and also irregular formations (resetid).

Some Swedish adjectives form Adj/N compounds with nouns.

If an English word A is ambiguous between an adjective and a noun, and N is a noun, then A + N could potentially be either an adjective-modified noun or a nominal compound. We disallow the second reading to limit spurious ambiguity.

Swedish compounds are non-trivial and our treatment is still not very satisfactory.

English rules:

`nbar_np_nbar_ComplexN` Compound nominal: proper noun + common noun

`nbar_nbar_nbar_ComplexN` Compound nominal: common noun + common noun

Swedish rules:

`nbar_nbar_CompoundingForm` Compounding form of common noun is base form

`nbar_nbar_infix1` Compounding form of common noun by adding affix (initial element of compound)

`nbar_nbar_infix2` Compounding form of common noun by adding affix (non-initial element of compound

`name_name_infix` Compounding form of proper noun by adding affix

`nbar_adjp_infix_nbar_ComplexN` Adjective + common noun compound

`nbar_nbar_infix_nbar_ComplexN` Common noun + common noun compound

`nbar_name_infix_nbar_ComplexN` Proper noun + common noun compound

`nbar_name_nbar_ComplexN` Proper noun + common noun compound with space

English feature (nbar and np):

`couldbeadjp` Noun is ambiguous between noun and adjective

Swedish features (np and nbar):

`compoundingform` Compounding form of word

`nn_infix` Initial/non-initial element of compound

Swedish feature (adjp):

`compoundingadj` Adjective that can form Adjective + common noun compounds

10.4.5 Modifier Constructions

10.4.6 -ing VP modifier

English allows use of an -ing VP as a VP or NP modifier

English rule:

`advp_vp_Ing` -ing VP as VP modifier

(Different feature settings on np_np_pp rule)

10.4.7 Extraction from of-PP

English has more restrictive rules governing extraction from NPs. Thus e.g. "Who_i is he the brother of e_i?" only works with "of". Swedish doesn't need a special rule here.

English rule:

`np_np_pp_OfPPGap` NP with dangling "of"

10.4.8 Phrasal Constructions

10.4.9 Time of day

Differences in types of expression handled:

English: three o'clock fourteen hundred hours

Swedish: klockan tre tre och tjugo tre femtiofem tre-tiden

10.4.10 Date expressions

Differences in types of expression handled:

English:
the twenty seventh of September
September twenty seventh
the twenty seventh
September twenty seven
twenty seven September
September the twenty seventh
September the twenty seventh nineteen ninety seven
Monday September the twenty seventh

Swedish:
tjugosjunde september
den tjugosjunde
tjugosjunde i nionde
den tjugosjunde september
den tjugosjunde i nionde

Chapter 11

French Coverage

Manny Rayner and Pierrette Bouillon

This chapter will describe the French version of the CLE, which, like the Swedish one, was created by intelligent manual adaptation of the original English version. The general style of exposition will also be similar to that adopted in the chapter on Swedish coverage; rather than recapitulate the many similarities between the French grammar and the English one from which it was derived, we will concentrate on the differences. In a final section, we briefly sketch an experimental Spanish grammar, which was developed by adaptation of the French system.

Throughout the chapter, we attempt to highlight the general nature of the adaptation process, and argue that it constitutes a general recipe that could be applied to other unification-based systems.

11.1 Introduction

We begin by summarizing the main capabilities of the French version of the system. The syntactic rule-set covers nearly all the basic constructions of the language, including the following: declarative, interrogative and imperative clauses; formation of YN and WH-questions using inversion, complex inversion and "est-ce que"; clitic pronouns; adverbial modification; negation; nominal and verbal PPs; complements to "être" and "il y a"; relative clauses, including those with "dont"; partitives, including use of "en"; passives; pre- and post-nominal adjectival modification, including comparative and superlative; code expressions; sentential complements and embedded questions; complex determiners; numerical expressions; date and time expressions; conjunction of most major constituents; and a wide variety of verb types, including modals and reflexives. There is a good treatment of inflectional morphology which includes all major paradigms. The coverage of the Spanish grammar is comparable in scope, though slightly less extensive. The French and Spanish versions of the CLE are both "reversible", and can be used for either analysis or generation.

We will describe the adaptation process in detail, and argue that it provides a fairly

general recipe for converting a grammar-based system for English into a corresponding one for a Romance language. The only components which required manual alteration were "rule" modules: hand-coded, unification-based descriptions of the grammar, lexicon, morphology etc. As noted in Alshawi (1992) Section 14.2.2 the effort involved in adapting a set of rule modules to a new language depends on how directly they refer to surface form; unsurprisingly, modules defining surface phenomena are the ones which require most work. When adapting the system to French and Spanish, the problems arose almost exclusively in connection with morphology and syntax rules. Other parts of the English system were adapted with little effort. In particular, the semantic rule-sets for English could be used for the new languages with only minimal changes.

The rest of the chapter is organized as follows. Section 11.2 describes the French morphology rules. Sections 11.3 and 11.4 describe the French and Spanish grammars. Section 11.5 concludes.

11.2 Morphology and spelling

In order to handle the more complex inflectional morphology of Romance and other European languages, a morphological processor based on feature-augmented two-level morphology was developed (Carter, 1995). This allows the complex spelling changes occurring in these languages to be handled quickly in both analysis and generation. Compilation of the full sets of two-level rules describing spelling changes and of production rules describing legal affix combinations takes of the order of a minute, allowing changes to the rules to be debugged relatively easily. Further flexibility is gained by not requiring the lexicon to be present at compile time (contrast Kaplan and Kay (1994)); thus the lexicon can be incremented and tested without any recompilation being required. Two-level spelling rules were also used to describe the inter-word effects that are particularly common in French.

The total number of rules required to describe inflectional morphology was around 75 for French and 50 for Spanish (inter-word rules being responsible for much of the difference). We concentrate here on the French phenomena, which are more complex.

11.2.1 Intra-word spelling changes

Intra-word spelling changes for French present several problems not encountered in English inflectional morphology. Some of these are technical in nature, and easily dealt with. In particular, French exhibits many multiple letter changes, e.g. "chameau+e" $\rightarrow$ *chamelle*, "peign+rai" $\rightarrow$ *peindrai*. For reasons explained in Carter (1995), these must be handled by a separate rule for each letter that changes, rather than one for the whole changed substring. Also, some changes can be optional. For example, the "y" in verbs such as "payer" can remain the same or change to "i" before silent "e": "pay+e" $\rightarrow$ either *paye* or *paie*. This phenomenon is rare or absent in English, but is handled easily by making the relevant spelling rule optional.

Less trivial problems, however, arise from the fact that spelling changes in French generally cannot be predicted from the surface form of the word alone. This means the application of the rules must be controlled; we do this by specifying feature constraints,

which must match between the rule and all morphemes it applies to. The following extended example describes our treatment of one of the most challenging cases.

Nouns, adjectives and verbs ending in "-et" or "-el" can either double the "t" or "l" before a silent "e" or change the prefinal "e" to "è": "cadet+e" → *cadette*, but "complet+e" → *complète*. The application of the spelling rules is therefore controlled by means of a feature `spelling_type`, with value `double` in the first case and `change_e_è` in the second.

This situation is further complicated by two facts. Firstly, the surface "èl" or "èt" of the verbs is ambiguous between a deep "el" or "et", and "él" or "ét". For example, we have *achète* ← "achet+e", but *affrète* ← "affrét+e". For this reason, we introduce a third value for `spelling_type`: `change_é_è`. "Affrét" has thus the feature `spelling_type=change_é_è`, "achet" `spelling_type=change_e_è` and "appel" `spelling_type=double`.

Secondly, the "e" that begins future and conditional endings sometimes affects preceding letters as if it were silent, and sometimes as if it were not. For example, "appel+erai" → *appellerai*, doubling the "l" just as in "appel+e" → *appelle*, where the final "e" actually is silent. However, "céd+erai" → *céderai*, not **cèderai* as would be expected from the silent-e behaviour "céd+e" → *cède*. To make this distinction, we use a feature `muet` ("silent") for specifying if the "e" in the suffix is silent, as "e" (`muet=y`), not silent, as "ez" (`muet=n`) or the "e" of the future or conditional tenses, for example "erai/erais" (`muet=fut_cond_e`). Then, we restrict the rule for doubling the consonant with the features `spelling_type=double, muet=y∨fut_cond_e`, and the one for "é" → "è" with the features `spelling_type=change_é_è,muet=y`.

11.2.2 Inter-word spelling changes

In English, inter-word spelling changes occur only in the alternation between "a" and "an" before consonant and vowel sounds respectively. In French, such changes are far more widespread and can be complex. However, they can be handled by judiciously specifying contexts in two-level rules and, in a few cases, by postulating non-obvious underlying lexical items. Some important cases are:

- The "e" in the function words "de", "je", "le", "me", "ne", "que", "se" and "te" is elided before (most) words starting in a vowel sound, except when the function word follows a hyphen: "le homme" → *l'homme*, "je ai" → *j'ai*, but "puis-je avoir" does not elide, so the elision rule specifies that the hyphen be absent from the context. "Ce" also elides when used as a pronoun ("ce est" → *c'est*, but when used as a determiner it takes the form "cet" before a vowel: *cet homme*. We therefore take the underlying form of the determiner to be "cet", which *loses* its "t" when followed by a consonant-initial word ("cet soir" → *ce soir*).

  Numerals do not allow elision either: "le onze" does not become **l'onze*. We therefore treat the lexical form as being "#onze", where "#" acts as an underlying consonant but is realised as a null. (Syntax plays a role here too: "le un" → *l'un* when is a determiner, but not when it is a numeral. Thus lexically we have "un" as determiner and "#un" as numeral).

- The very common preposition/article combinations “de”/“à” and “le”/“les”: “de le” → *du*, “à les” → *aux*, etc. These contractions span constituent boundaries (we view *du vol* as being syntactically [PP de [NP le vol]]) so need to be treated as spelling effects. Also, vowel elision takes precedence: “de le homme” → *de l’homme*, not **du homme*.

- Hyphens between verbs and clitic pronouns are treated as lexical items in our grammar. They are realised as *-t-* when preceded by “a” or “e” and followed by “e”, “i” or “o”: “va - il” → *va-t-il*, but “vont - ils” → *vont-ils*. Hyphens joining nouns or names are treated as different lexical items not subject to this change: “les vols Atlanta - Indianapolis” does not involve introduction of “t”.

11.3 French syntax

When comparing the French and English grammars, there are two types of objects of immediate interest: *syntax rules* and *features*. Looking first at the rules themselves, about 80% of the French syntax rules are either identical with or very similar to the English counterparts from which they have been adapted. Of the remainder, some rules (e.g those for date, time and number expressions) are different, but essentially too trivial to be worth describing in detail. Similar considerations apply to features.

We will concentrate our exposition on the rules and features which are both significantly different, and possess non-trivial internal structure. Examining the grammar, we find that there are three large interesting groups of rules and features, describing three separate complexes of linguistic phenomena: question-formation, clitic pronouns and agreement. As we have argued previously (Rayner and Bouillon, 1995) all of these are rigid and well-defined types of construction which occur in all genres of written and spoken French. It is thus both desirable and reasonable to attempt to encode them in terms of feature-based rules, rather than (for instance) expecting to derive them as statistical regularities in large corpora. In Sections 11.3.2, 11.3.1 and 11.3.3, we describe how we handle these key problems.

11.3.1 Question-formation

We start this section by briefly reviewing the way in which question-formation is handled in the English CLE grammar. There are two main dimensions of classification: questions can be either WH- or Y-N; and they can use either the inverted or the uninverted word-order. Y-N questions must use the inverted word-order, but both word-orders are permissible for WH-questions. The phrase-structure rules analyse an inverted WH-question as constituting a fronted WH+ element followed by an inverted clause containing a gap element. The feature `inv` distinguishes inverted from uninverted clauses. The following examples illustrate the top-level structure of Y-N, unmoved WH- and moved WH-questions respectively.

[Does he love Mary]$_{S:[inv=y]}$

[Who loves Mary]$_{S:[inv=n]}$

[[Whom]$_{NP}$ [does he love []$_{NP}$]$_{S:[inv=y]}$]

The French rules for question formation are structurally fairly similar to the English ones. However, there are several crucial differences which mean that the constructions in the two languages often differ widely at the level of surface form. Two phenomena in particular stand out. Firstly, English only permits subject-verb inversion when the verb is an auxiliary, or a form of "have" or "be"; in contrast, French potentially allows subject-verb inversion with any verb. For this reason, English question-formation using auxiliary "do" lacks a corresponding construction in French.

Secondly, French permits two other common question-formation constructions in addition to subject-verb inversion: prefacing the declarative version of the clause with the question particle "est-ce que", and "complex inversion", i.e. fronting the subject and inserting a dummy pronoun after the inverted verb. In certain circumstances, primarily if the subject is the pronoun "ça", it is also possible to form a non-subject WH-question out of a fronted WH+ phrase followed by an uninverted clause containing an appropriate gap. We refer to this last possibility as "pseudo-inversion".

If the subject is a pronoun, only inversion and the "est-ce que" construction are allowed; if it is *not* a pronoun, only the "est-ce que" construction and complex inversion are valid. In addition, a subject pronoun following an inverted verb needs to be linked to it by a hyphen, which can be realised as a "-t-" (see Section 11.2). Figure 11.1 presents examples illustrating the main French question constructions.

Modification of the English syntax rules to capture the basic requirements so far is quite simple. In our grammar, we added three extra rules to cover the "est-ce que", complex-inversion and pseudo-inversion constructions: the second of these rules combines the complex-inverted verb with the following dummy pronoun to form a verb, in essence treating the dummy pronoun as a kind of verbal affix. A further rule deals with the hyphen linking an inverted verb with a following subject.

With regard to the feature-set, the critical change involves the `inv` feature. In English, as we saw, this feature had two possible values, `y` and `n`. In French, the corresponding feature has five values: `inverted`, `uninverted`, `est_ce_que`, `complex` and `pseudo`, distinguishing clauses formed using the different question-formation constructions. (It is important to note, though, that the semantic representation of the clause is the same irrespective of its inversion-type). To enforce the restrictions concerning combinations of inversion-type and subject form, we also added a new clausal feature which distinguished clauses in which the subject is a pronoun.

The attractive aspect of this treatment is that the remaining English rules used for question-formation can be retained more or less unchanged. In particular, the English semantic rules can still be used, and produce QLF representations with similar form.

It would almost be true to claim that the above constituted our entire treatment of French question-formation. In practice, we have found it desirable to add a few more features to the grammar in order to block infelicitous combinations of the inversion rules with certain commonly occurring lexical items. It is possible that the effect of these features could be achieved equally well by statistical modelling or other means, but we describe them here for completeness:

Restrictions on use of "est-ce que": Question-formation with "est-ce que" is strongly

Y-N, inversion:
Aime-t-il Marie?

Y-N, "est-ce que":
Est-ce que Jean aime Marie?

Y-N, complex inversion:
Jean aime-t-il Marie?

WH, subject question, no inversion:
Quel homme aime Marie?

WH, inversion:
Quelle femme aime-t-il?

WH, "est-ce que":
Quelle femme est-ce que Jean aime?

WH, complex inversion:
Quelle femme Jean aime-t-il?

WH, pseudo-inversion:
Combien ça coûte?

Figure 11.1: Main French question constructions

dispreferred when the main verb is a clause-final occurrence of “être”, or existential “avoir” (as in “il y a”). For example:

?Quand est-ce que le prochain vol est?
?Combien de vols est-ce qu’il y a?

We enforce this by adding a suitable feature to the verb category.

Fronting of “heavy” NPs: Most languages prefer not to front “heavy” NPs, and this dispreference is particularly strong in French. We have consequently added an NP feature called `heavy`, which has the value `y` on NPs containing PP and VP post-modifiers. Thus for example generation of

Quels vols en partance de Dallas y a-t-il?

is blocked, but the preferable

Quels vols y a-t-il en partance de Dallas?

is permitted.

Inverted subject NPs: Occurrence of some pronouns (in particular “cela”, and “ça”) is strongly dispreferred in inverted subject position. A binary feature enforces this as a rule, for example blocking

Combien coûte ça pour aller à Boston?

but instead permitting

Combien ça coûte pour aller à Boston?

11.3.2 Clitics

The most difficult technical problems in adapting an English grammar to a Romance language are undoubtedly caused by clitic pronouns. In contrast to English, certain proform complements of verbs do not appear in their normal positions; instead, they occur adjacent to the main verb, and possibly joined to it by a hyphen. The position of the clitics in relation to the verb (pre- or post-verbal) is determined by the mood of the verb, and whether or not the verb is negated. If two or more clitics are affixed to the verb, their internal order is determined by their surface forms. Several attempts to account for the above and other data have previously been described in the literature e.g. (Grimshaw, 1982; Bès and Gardent, 1989; Estival, 1990; Miller and Sag, 1995); we have in particular been influenced by the last of these.

Although the underlying framework is very different from the HPSG formalism used by Miller and Sag, our basic idea is the same: to treat “clitic movement” by a mechanism similar to the one used to handle WH movement. More specifically, we introduce two sets of new rules. The first set handles the “surface” clitics. They define the structure of the verb/clitic complex, which we, like Estival, regard as a constituent of category V composed of a main verb and a “clitic-list”. A second set of “gap” rules defines empty constituents of category NP or PP, occurring at the notional “deep”

positions occupied by the clitics. Thus, for example, on our account the constituent structure of "Est-ce que vous le voulez?" will be

[Est-ce que [vous$_{NP}$ [le voulez]$_V$ []$_{NP}$]$_S$]$_S$

where the "gap" NP category represents the notional direct object of "voulez", realised at surface level by the pre-verbal clitic "le".

To make this work, we add an extra feature, `clitics`, to all categories which can participate in clitic movement: in our grammar, these are V, VP, S, NP and PP. The `clitics` feature is used to link the cliticised V constituent and its associated clitic gap or gaps. We have found it convenient to define the value of the `clitics` feature to be a bundle of five separate sub-features, one for each of the five possible clitic-positions in French. Thus for instance the second-position clitics "le", "la" and "les" are related to object-position clitic gaps through the second sub-feature of `clitics`; the fourth-position "y" clitic is related to its matching PP gap through the fourth sub-feature; and so on. The linking relation between a clitic-gap and its associated clitic is formally exactly the same as that obtaining between a WH-gap and its associated antecedent, and can if desired be conceptualized as a type of coindexing.

The `clitics` feature-bundle is threaded through the grammar rule which defines the structure of the list of clitics associated with a cliticised verb, and enforces the constraints on ordering of surface clitics. These constraints are encoded in the lexical entry for each clitic.

This basic framework is fairly straight-forward, though a number of additional features need to be added in order to capture the syntactic facts. We summarize the main points:

Position of surface clitics: Clitics occur post-verbally in positive imperative clauses, otherwise pre-verbally. The clitic-list constituent consequently needs to share suitable features with the verb it combines with.

Surface form of clitics: The first- and second-person singular clitics are realised differently depending on whether they occur pre- or post-verbally: for example "Vous me réservez un vol" versus "Réservez-moi un vol". Moreover, "me" and "te" are first-position clitics (e.g. "Vous me les donnez"), while "moi" and "toi" are third-position ("Donnez-les-moi"). This alternation is achieved simply by having separate lexical entries for each form. The entries have different syntactic features, but a common semantic representation.

Special problems with the "en" clitic: The most abstruse problems occur in connection with the "en" clitic, and are motivated by sentences like

Combien en avez-vous?

Here, our framework seems to dictate a constituent structure including three gaps, viz:

[Combien [[en avez]$_V$ [vous$_{NP}$ [[]$_V$ [[]$_{NP}$ []$_{PP}$]$_{NP}$]]$_S$]$_S$]$_S$

in which the V gap links to “avez”, the NP gap to “combien”, and the PP gap to “en”. The specific difficulty here is that the “en” PP gap ends up as an NP modifier (it modifies the NP gap). Normally, however, PP modifiers of NPs cannot be gaps, and the above type of construction is the only exception we have found.

Rather than relax the very common `NP` $\rightarrow$ `NP PP` rule to permit a gap PP daughter, we introduce a second rule of this type which specifically combines certain NPs, including suitable gaps resulting from WH-movement, and an “en” clitic gap. A feature, `takespartitive`, picks out the NPs which can participate as left daughters in this rule.

11.3.3 Agreement

Although grammatical agreement is a linguistic phenomenon that plays a considerably larger role in French than in English, the adjustments needed to the lexicon and syntax rules are usually obvious. For instance, a feature has to be added to the both daughters of the rule for pre-nominal adjectival modification, to enforce agreement in number and gender. In nearly all cases, this same procedure is used. A feature called `agr` is added to the relevant categories, whose value is a bundle representing the category’s person, number and gender, and the `agr` feature is shared between the categories which are required to agree.

There are however some instances where agreement is less trivial. For example, the subject and nominal predicate complement of “être” may occasionally fail to agree in gender, e.g.

La gare est le plus grand bâtiment de la ville.

However, if the predicate complement is a pronoun (“lequel”, “celui-ci”, “quel”[1]...) agreement in both gender and number is obligatory: thus for instance

Quel/*quelle/*quels est le premier vol.

It would be most unpleasant to duplicate the syntax rules, with separate versions for the pronominal and non-pronominal cases. Instead, we add a second agreement feature (`compagr`) to the NP category, which is constrained to have the same value as `agr` on pronominal NPs; subject/predicate agreement can then use the `compagr` feature on the predicate, getting the desired behaviour.

Similar considerations apply to the rule allowing modification of a NP by a “de” PP. In general, there is no requirement on agreement between the head NP and the NP daughter of the PP. However, for certain pronominal NP (“lequel”, “l’un”, “chacun”) gender agreement is obligatory, e.g.

lequel/*laquelle de ces vols
laquelle/*lequel de ces dates

[1]Most French grammars regard “quel” as an adjective, but for semantic reasons we have found it more convenient to treat it as a pronoun in this type of construction and as a determiner in expression like “quel vol”.

This is dealt with correspondingly, by addition of a new agreement feature specific to the NP → NP PP rule.

11.4 Spanish syntax

This section briefly describes the interesting features of the Spanish syntactic rule-set. In general, the Spanish rules were distinctly simpler than the French ones. With a few exceptions noted below (in particular, prodrop), the current Spanish syntax rules are essentially a slightly modified subset of the French ones. Despite this, they give very adequate coverage of the ATIS domain, the only in which they have so far been seriously tested. In a little more detail:

Question-formation: The Spanish rules for question-formation are similar to, but less elaborate than the French ones. Subject-verb inversion is allowed with any subject; there is no restriction that it be pronominal. There are no constructions corresponding to "est-ce que" or complex inversion. When the inverted subject is a pronoun, it does not require a preceding hyphen linking it to the verb.

Clitics: The Spanish clitic system is also considerably simpler than the French one. There are fewer clitics; in particular, there are no clitics corresponding to the French "y" and "en", which as we saw in Section 11.3.2 above gave rise to many of the difficult problems in French.

Postverbal clitics are affixed directly to the verb, rather than being joined by hyphens. Since CLE morphosyntactic rules have a uniform format (Alshawi, 1992, Section 3.9), this only involved moving the relevant syntax rules to the morphology rule file.

Phrasal rules: The rules for Spanish numbers, dates and times are substantially different from the French ones, and those for dates in particular needed to be rewritten more or less from scratch. The issues involved are however straight-forward.

Also, the form of the Spanish superlative adjective is slightly different: the postnominal superlative adjective has no extra article, e.g. "le vol le [plus cher] versus "la plaza [menos cara]". The necessary adjustments are again simple.

Relative clauses: A less trivial difference involves relative clauses. In Spanish, the main verb of the relative clause must be in the subjunctive mood if it modifies an argument of a verb in the imperative mood. Thus for example

> Which is the first flight that serves a meal?
> → Cuál es el primer vuelo que sirve una comida?

("sirve" = present indicative), but

> Show me flights that serve a meal!
> → Enséñeme los vuelos que sirva una comida

("sirva" = present subjunctive). Handling this alternation correctly involves trailing an extra feature through many grammar rules, so as to link the main verb in the relative clause to the main verb in the clause immediately above it.

Prodrop: The second substantial change required when adapting the French grammar to Spanish was necessitated by the prodrop rule: Spanish, unlike French, permits and indeed encourages omission of the subject when it is a pronoun. Perhaps surprisingly, prodrop in fact only resulted in a few divergences between the Spanish and French grammars. A new syntax rule of the form $\mathsf{S} \rightarrow \mathsf{VP}$ was added (it is in fact a slightly modified version of the French imperative-formation rule). The associated semantic rule fills in a representation of the omitted clausal subject from the main verb; to make this possible, the semantic entries for inflected verbs are all modified to contain an extra feature encoding the possible prodrop subject. The details are straight-forward.

11.5 Conclusions

The preceding sections describe in essence all the changes we needed to make in order to adapt a substantial English language processing system to French and Spanish. We have perhaps presented some of the details in a more compressed form than we would ideally have wished, but nothing important has been omitted. Creation of a good initial French version required about five person-months of effort; after this, the Spanish version took only about two person-months. We do not believe that we were greatly aided by any special features of the Core Language Engine, other than the fact that it is a well-engineered piece of software based on sound linguistic ideas. Our overall conclusion is that an English-language system conforming to these basic design principles should in general be fairly easy to port to Romance languages.

Chapter 12

Transfer and Robust Translation

Manny Rayner, Pierrette Bouillon, Ivan Bretan and Mats Wirén

12.1 Introduction

When comparing speech translation and text translation, there are several obvious differences in the requirements posed by the task. As usual, some of the requirements conflict, resulting in implementation tensions. We will begin the chapter on transfer by noting what we regard as the most important requirements and tensions.

One important difference between speech translation and text translation is that speech translation poses stronger demands on quality of output. If output is not good enough, people frequently have difficulty understanding what has been said. There is no possibility of the pre- or post-editing which nearly all text translation systems rely on. Quite apart from the problem of generating natural-sounding speech, it is also necessary to ensure that the translated text sent to the speech synthesizer is itself of sufficient quality. A high-quality translation must fulfill several criteria: in particular, it should preserve the meaning of the original utterance, be grammatical, and contain correct word-choices.

However, the demand for high output quality conflicts with a second requirement: robustness is more important too, since spoken language is inherently noisier (in several senses) than written language. So a successful spoken language architecture needs to address the question of processing input which may be malformed in any one of a number of ways, and producing as sensible a result as possible rather than simply giving up.

Finally, we have a third important requirement. Speech translation is normally carried out in an interactive context, rather than being performed off-line like most text translation. Thus processing speed is also more important for speech than for text.

Clearly, it is not possible to focus on all these issues at once without introducing some kind of compromise. Trying for greater robustness inevitably degrades output

quality, and increased speed must adversely affect all other aspects of processing. The important thing is to attempt to find a good trade-off between the various competing factors.

Our basic recipe has been to combine two different types of transfer. When possible, we translate using a sophisticated method, which in accordance with most of the rest of the system combines unification-based rules and numerical preferences. The sophisticated method performs transfer at the level of Quasi Logical Form, and is described in Section 12.2. We will refer to this method as "QLF transfer".

As with most rule-based methods, there are cases where QLF transfer fails to produce a result, due to noise in the input utterance, holes in the rule-sets, or both. In this case, we fall back on a simpler and more robust transfer method, which operates at the level of rules mapping lists of surface lexical items into lists of surface lexical items. We refer to this method as "word-to-word transfer", or "WW transfer". The two methods, QLF transfer and WW transfer, are applied in parallel in a bottom-up mode, and the results are combined to attempt to produce as good a translation as possible of the full utterance. How this is done is explained in Section 12.3.

12.2 QLF-based transfer

This section will describe the QLF-based transfer component in detail. We begin in Section 12.2.1 with some introductory remarks motivating the basic design. In particular, we describe our reasons for structuring the QLF transfer component as a hybrid architecture, which combines rules and numerical preferences. We also describe the basic structure of the transfer component, and give examples of transfer rules and transfer preferences. Section 12.2.2 then explains, with the help of a number of examples, how rules and preferences combine to solve several types of non-trivial transfer problems. In the current version of the system, the numerical transfer preferences are coded by hand. Section 12.2.3 briefly describes some experiments in which we have attempted to derive them (semi-)automatically via supervised training.

The last three sections deal with some more peripheral topics. Section 12.2.4 describes a "packing" technique used to increase the efficiency of the transfer process. Section 12.2.5 describes the pre- and post-transfer phases. Finally, Section 12.2.6 looks at the problem of correctly handling logical variables in QLF transfer.

12.2.1 Introduction

The basic design philosophy of the SLT project has been to build a framework which is theoretically clean, on the usual grounds that this makes for a system that is portable and easy to scale up. We have attempted to subsume as much as possible of the transfer component under two standard paradigms: *unification-based language processing* and the *noisy-channel statistical model*. The unification-based part of the system encodes domain-independent contrastive grammatical rules; for each source-language word or grammatical construction covered by the system, it describes the possible target-language translations. When the rules permit more than one potentially valid

translation, the preference component is used to rank them in order of relative plausibility. The next two paragraphs give some examples to motivate this division of knowledge sources.

The simplest examples of transfer rules are those used to translate individual words; here it is immediately clear that many words can be translated in several ways, and thus that more than one rule will often apply. For instance[1], the English preposition *"on"* can be translated as any of the French prepositions *"avec"* (*fly to Boston on Delta* $\rightarrow$ *aller à Boston avec Delta*); *"sur"* (*information on ground transportation* $\rightarrow$ *des renseignements sur les transports publics*); *"à bord de"* (*a meal on that flight* $\rightarrow$ *un repas à bord de ce vol*); *"pour"* (*the aircraft which is used on this flight* $\rightarrow$ *l'avion qu'on utilise pour ce vol*); or omitted and replaced by an implicit temporal adverbial marker (*leave on Monday* $\rightarrow$ *partir le lundi*). In each of these cases, the correct choice of translation is determined by the context.

To take a slightly more complex case, which involves some grammar, there are a number of transfer rules that list possible ways of realizing the English compound nominal construction in French. Among these are adjective + noun (*economy flight* $\rightarrow$ *vol économique*); noun + PP (*arrival time* $\rightarrow$ *heure d'arrivée*; *Boston ground transportation* $\rightarrow$ *transports publics à Boston*); or in special cases simply a compound noun (*Monday morning* $\rightarrow$ *lundi matin*). Again, the individual lexical items and the context determine the correct rule to use.

Experience has shown that it is relatively simple to write the context-independent rules which list sets of choices like the ones above. It is however much more difficult to use rules to specify the context in which each particular choice is appropriate. Moreover, the correct choice is frequently domain-dependent; thus the rules will need to be rewritten if the system is ported to a new application. For these reasons, statistically trained machine translation architectures have recently been receiving a great deal of attention. Some researchers (notably those in the IBM CANDIDE project, (Brown *et al.*, 1990) have even gone so far as to claim that statistical techniques are sufficient on their own. Our view is that this is at best unnecessary. Since many aspects of language (for instance, agreement and question-formation in French) appear to be regular and readily describable by rules, it seems more logical to use a mixture of rules and statistics; it is in this sense that we have a *hybrid* transfer model (see Brown it et al. 1992, Carbonell 1992, Grishman and Kosaka 1992). In the remainder of this section, we will describe the basic functionality of the rule-based and preference aspects of the QLF transfer component.

Transfer rules are written in a minimal unification-based formalism, which allows rules of two kinds: *simple* and *recursive*. Simple rules map source-language QLF fragments onto their target-language counterparts. For example, rules (1)–(3) below are of this type.

```
(1)   trule([eng, fre],
      flight1 >= vol1).

(2)   trule([eng, fre],
```

[1]We will take most of our examples in this chapter from the English/French transfer pair, on the grounds that it is linguistically rather more interesting than English/Swedish.

```
      on1     >= sur1)

(3)   trule([eng, fre],
      on1     >= avec1)
```

(1) states that the English QLF constant `flight1` maps into the French QLF constant `vol1`; (2) and (3) that the English QLF constant `on1` can map into either of the French QLF constants `sur1` or `avec1`. Note that the choice between the two different ways of translating `on1` is not expressed in rules (2) and (3) themselves, but is left to the transfer preferences. Simple transfer rules compile into unit clauses: thus (1) compiles into the unit clause (1a)

```
(1a)  transfer([eng,fre], flight1, vol1).
```

Recursive transfer rules use *transfer variables* to express transfer of a QLF expression in terms of transfer of one or more of its sub-expressions. For example, (4) is a simplified version of the rule which can be used to translate an English compound nominal (e.g. "arrival time") into a French noun/PP combination ("heure d'arrivée"). The QLF constant `nn` on the left-hand side represents the compound nominal relationship obtaining between `noun1` and `noun2`, and is translated into an expression on the right-hand side representing a French preposition-sense.

```
(4)   trule([eng, fre],
      form(tr(relation,nn),
           tr(noun1),
           tr(noun2))
      >=
      [and, tr(noun2),
            form(prep(tr(relation)),
                 tr(noun1))]).
```

Transfer variables are terms of the form `tr(Id)` or `tr(Id,Pat)`, where `Id` is an identifier and `Pat` is an optional pattern. The expression matching the transfer variable tagged `Id` on the left-hand side of the rule is transferred into the expression matching the variable with the same identifier on the right-hand side; if a pattern is supplied on either side, the matching expression on that side is unified with the pattern. So (4) compiles to the Horn-clause (4a):[2]

```
(4a)  transfer([eng,fre],
               form(nn, Y, Z),
               [and, Z1,
                     form(prep(X1), Y1)])
      <-
      transfer([eng,fre], nn, X1),
      transfer([eng,fre], Y, Y1),
      transfer([eng,fre], Z, Z1).
```

[2]We will find it convenient to write Horn-clauses in a "Prolog-style" notation, with implicit wide-scope universal quantification over free variables.

Moving on to the numerical prefences, the basic idea is inspired by the noisy-channel statistical model of translation described in (Brown *et al.* 1990). The plausibility of a new candidate transfer is defined to be a real number, calculated as a weighted sum of two contributions: the *transfer rule score*, and the *target language model score*. The first of these represents the relative plausibility of the rules used to make the transfer, and the second the plausibility of the target QLF produced.

The transfer rule score and the target language model score are computed using the same method; for clarity, we first describe this method with reference to transfer rules. The transfer rule score for the bag of transfer rules used to produce a given target QLF is a sum of the *discriminant scores* for the individual transfer rules. The discriminant score for a rule R summarizes the reliability of R as an indicator that the transfer is correct or incorrect. The intent is that transfer rules which tend to occur more frequently in correct transfers than incorrect ones will get positive scores; those which occur more frequently in incorrect transfers than correct ones will get negative scores.

The target language model score is defined similarly. The first step is to extract a bag of "semantic triples" (see Section 7.3.3) from each possible transferred QLF in the training corpus, following which each individual triple is assigned a discriminant score. Semantic triples encode grammatical relationships between head-words; we have generalized the original definition used in the SLT-1 prototype (Alshawi and Carter, 1994) to include relationships involving determiners, since these are important for transfer. Thus for example the normal reading of the English sentence

Show flights with a stop.

would include the triples

```
(show,obj,flight)    (show,obj,bare_plur)
(bare_plur,det,flight) (flight,with,stop)
(flight,with,a)        (a,det,stop)
```

Ideally, transfer discriminant scores should be derived according to the kinds of method described in Section 6.4, using a statistical computation based on a training corpus of judged candidate translations. We have experimented with this idea (see Section 12.2.3), and in the long run it must clearly be the correct approach. However, due to technical problems explained below, these experiments so far produce results no better than those we get using hand-coded transfer preferences. The current version of the system consequently still includes hand-coded transfer preferences.

12.2.2 Combining transfer rules and transfer preferences

This section will describe examples of non-trivial translation problems from the ATIS domain, and describe how the QLF transfer component deals with them. We were interested to discover that even a domain as simple as ATIS actually contains many quite difficult transfer problems. We will begin by giving examples where it is fairly clear that the problem is essentially grammatical in nature, and thus primarily involves the rule-based part of the system; later, we give examples where the problem mainly

involves the preference component, and examples where both types of knowledge are needed.

An obvious case of a grammatical phenomenon is agreement, which is considerably more important in French than in English; the rules for agreement are rigid and well-defined, and easy to code in a feature-based formalism. Quite frequently, however, they relate words which are widely separated in the surface structure, which makes them hard to learn for surface-oriented statistical models. For example, there are many instances in ATIS of nouns which in French are postmodified both by a PP and by a relative clause, e.g.

Flights from Boston to Atlanta leaving before twelve a m
→ *Les vols de Boston à Atlanta qui partent avant midi*

Here, the verb *partent* has to agree in number and person with the head noun *vols*, despite the gap of five surface words in between.

Many problems related to word-order also fall under the same heading, in particular those relating to question-formation and the position of clitic pronouns. For example, French YN-questions can be formed in three ways: by inversion of subject and main verb, by prefacing the declarative version of the clause with the question particle *est-ce que*, or by "complex inversion", fronting the subject and inserting a dummy pronoun after the inverted verb. If the subject is a pronoun, only the first and second alternatives are allowed; if it is *not* a pronoun, only the second and third are valid. Thus for example

Does it leave after five p m?
→ *Part-il après dix-sept heures?*
→ *Est-ce qu'il part après dix-sept heures?*
→ **Il part-il après dix-sept heures?*

Does that flight serve meals?
→ **Sert ce vol des repas?*
→ *Est-ce que ce vol sert des repas?*
→ *Ce vol sert-il des repas?*

Embedded questions constitute another good example of a mainly grammatical problem. Just as in English, French embedded questions normally have the uninverted word-order, e.g.

Tell me ***when these flights arrive in Boston***
→ *Dites-moi* ***quand ces vols arrivent à Boston***

However, if the main verb is *être* with an NP complement, the inverted word-order is obligatory, e.g.

Tell me ***what the cheapest fares are***
→ *Dites-moi* ***quels sont les tarifs les moins chers***
→ **Dites-moi* ***quels les tarifs les moins chers sont***

In ATIS, embedded questions occur in about 1% of all corpus sentences; this makes them too frequent to ignore, but rare enough that a pure statistical model will probably

have difficulties finding enough training examples to acquire the appropriate regularities. The relevant facts are however quite easy to state as grammatical rules. Moreover, they are domain-independent, and can thus be reused in different applications.

In contrast, there are many phenomena, especially involving word-choice, which are hard to code as rules and largely domain- and application-dependent. As mentioned earlier in Section 12.1, the translation of prepositions and determiners is most frequently determined on collocational grounds; in our framework, this means that the information used to decide on an appropriate translation is primarily supplied by the transfer preferences. We will now describe in more detail how the idea works in practice.

Recall that the preference score for a given transfer candidate is a weighted sum of a channel contribution (discriminants on transfer rules) and a target language model score (discriminants from target language semantic triples). The transfer rule discriminants make transfer rules act more or less strongly as defaults. If a transfer rule R is correct more often than not when a choice arises, it will have a positive discriminant, and will thus be preferred if there is no reason to avoid it. If use of R produces a strong negative target-language discriminant, however, the default will be overridden.

Let us look at some simple examples. The English indefinite singular article *"a"* can be translated in several ways in French, but most often it is correct to realise it as an indefinite singular (*"un"* or *"une"*). The discriminant associated with the transfer rule that takes indefinite singular to indefinite singular is thus fairly strongly positive. There are however several French prepositions which have a strong preference for a bare singular argument; for instance, *"flights without a stop"* is almost always better translated as *"les vols sans escale"* than *"les vols sans une escale"*. In cases like these, the *a*-to-*un* rule will be wrong, and the less common rule that takes indefinite singular to bare singular will be right. This can be enforced if the negative discriminant associated with the semantic triple

```
(vol, sans, indef_sing)
```

has a higher absolute value than the positive discriminant associated with *a*-to-*un*, and is able to overrule it.

Similar considerations apply to prepositions. In the ATIS domain, most prepositions have several possible translations, none of which are strongly preferred. For example, the channel score discriminants associated with the transfer rules *on*-to-*sur* and *on*-to-*avec* both have low absolute values; the first is slightly negative, and the second slightly positive. Target language triples associated with these prepositions are however in general more definite: the triples

```
(aller avec <airline>)
(renseignement sur transports)
```

are both strongly positive, while

```
(aller sur <airline>)
(renseignement avec transports)
```

are strongly negative. The net result is that the target language contribution makes the decision, and as desired we get *"fly on Delta"* and *"information on flights"* going to

"aller avec Delta" and *"des renseignements sur les vols"* rather than *"aller sur ..."* and *"des renseignements avec ..."*.

In general, a combination of rules and collocational information is needed to translate a construction. A good example is the English implicit singular mass determiner, which is common in ATIS. Grammatical rules are used to decide that there is a singular mass determiner present, following which the correct translation is selected on collocational grounds. An elementary French grammar will probably say that the normal translation should either be the French partitive singular determiner, e.g.

I drink ***milk***
→ *Je bois* ***du lait***

or else the definite singular, e.g.

I like ***cheese***
→ *J'aime* ***le fromage***

In the ATIS domain, it happens that the nouns which most frequently occur with mass singular determiner are *"transportation"* and *"information"*, both of which are conventionally singular in English but plural in French. Because of this, neither of the standard rules for translating mass singular gets a strong positive discriminant score, and once again the target language model tends to make the decision. For instance, if the head noun is *"transportation"*, it is most often correct to translate the mass singular determiner as a definite plural, e.g.

Show me ***transportation*** *for Boston*
→ *Indiquez-moi* ***les transports*** *pour Boston*

This is captured in a strong positive discriminant score associated with the target language triple

```
(def_plur, det, transport)
```

Note that the translation *"transportation"* to *"les transports"* is only a preference, not a hard rule; it can be overridden by an even stronger preference, such as the preference against having a definite plural subject of an existential construction. So we have e.g.

Is there ***transportation*** *in Boston?*
→ *Y a-t-il* ***des transports*** *à Boston?*
→ **Y a-t-il* ***les transports*** *à Boston?*

12.2.3 Training transfer preferences

This section describes preliminary experiments designed to test the possibility of deriving transfer preferences using a supervised training process. A set of 2000 ATIS utterances was used, randomly selected from the subset of the ATIS corpus consisting of A or D class[3] utterances of length up to 15 words, which had not previously been

[3]This means roughly that the sentence represented a valid inquiry to the database, either alone or in the context in which it was uttered.

examined during the development of the English/French version of SLT. Utterances were supplied in text form, i.e. the speech recognition part of the system was not tested here.

Each utterance was analysed using the English language version of the CLE, and for the 1847 sentences where at least one QLF was produced the most plausible QLF was selected using the normal preference methods. This was then submitted to the QLF transfer component, and a set of transfer candidates produced. A simple set of hand-coded transfer preferences was applied, and one French surface string was generated for each of the five highest-scoring transfer candidates. A native French speaker fluent in English judged each generated string as being either an acceptable or an unacceptable translation of the source utterance. Translations were only regarded as acceptable if they were fully grammatical, preserved the meaning of the source utterance, and used a stylistically natural choice of words. The judging process took approximately eight hours, averaging three seconds per source/target pair.

The annotated N-best transfer corpus was then used to train a new set of preferences using a variant of the method described in (Alshawi and Carter 1994); the corpus was divided into five equal pieces, each fifth being held out in turn as test data with the remaining four-fifths used as training. Finally, the derived preferences were tested for accuracy. Of the 1847 transfer sets, there were 1374 for which at least one acceptable transfer was in the top five candidates[4]. The trained transfer preferences selected an acceptable candidate in 1248 of these 1374 cases (91%); in contrast, random choice among the top five gave a baseline score of 826 acceptable transfers, or 60%.

Although these figures are encouraging, they are in fact not substantially better than those we have obtained from the hand-coded preferences; the current prototype consequently still uses the hand-coded preferences. We have attempted to analyse the reasons for this comparative failure, and are currently experimenting with a methodology which is intended to combine the good points of hand-coding and supervised training. The rest of the section briefly summarizes our experiences to date.

As explained above, it is clearly *possible* to train translation preferences from a corpora of good and bad examples of translation pairs. However, the task of producing such corpora was found to be unnecessarily labour intensive. If a sentence is judged bad because, for example, a certain adjective could never go with a particular noun, the judge wants to be able to state this, and not be presented with the same phenomenon in fifty other sentences. The new judging procedure therefore allows judges to alter the discriminant scores directly, and provides immediate feedback on the results of such changes.

The basic judging loop has the following stages:

1. The system picks the top source QLF from a relevant corpus file. (e.g. if translating from English to French this would be the first QLF from a file containing English QLFs formed after pre-transfer).

2. The system calculates its preferred translation. If this has already been judged

[4]There were a further 246 sets in which at least one candidate translation was produced; in most of these cases, the best translation was comprehensible and grammatically correct, but was rejected on stylistic grounds.

to be good, the system returns to (1). Otherwise, the user is presented with the original English sentence and the French translation.

3. The user judges the translation as good or bad. If good, the system's current preferences are giving the right results, so we go to the next QLF (i.e. return to step 1).

4. If the translation is judged bad, the system provides 4 more translations. The user is then asked which of these are the best translations. The system then presents the discriminants which occurred in the good translations versus those that appeared in the bad. This usually allows the user to quickly spot the relevant discriminant which needs changing.

5. The user changes one or more discriminants

6. The system recalculates its preferred translation, and presents it to the user. If this is OK the user goes to (1), otherwise back to (4).

After finishing a loop, it is good practice to go back into the loop to ensure that decisions later on have not affected the results for earlier translations. The system always checks existing judgement files (and adds to these during the judging procedure), so the user is never asked to make an unnecessary judgement.

12.2.4 Transfer packing

As already indicated, the basic philosophy of the transfer component is to make the transfer rules context-independent, and let the results be filtered through the numerical transfer preferences. The positive side of this is that the transfer rules are robust and simple to understand and maintain. The negative side is that non-deterministic transfer choices multiply out, giving a combinatoric explosion in the number of possible transferred QLFs.

To alleviate this problem, transferred QLFs are *packed*, in the sense of Tomita (1986); lexical transfer ambiguity is left "unexpanded", as a locally ambiguous structure in the target QLF. It is possible to compute preference scores efficiently on the packed QLFs, and only unpack the highest-scoring candidates; this keeps the transfer phase acceptably efficient even when several thousand transferred QLFs are produced.

The following example illustrates how transfer packing works. The source utterance is

flights on Monday

and the packed transferred QLF (in slightly simplified form) is:

```
elliptical_np(
  term(/|\(1,[def_plur,
              indef_plural,
              bare_plur]),
       C^[and,
          [vol1,C],
```

```
          form(prep(/|\(2,[a_bord_de,
                           temporal_np,
                           sur,
                           pour,
                           avec])),
               term(/|\(3,[def_sing,
                           bare_sing])
                    E^[lundi1,E]))]))
```

This contains three lexical transfer ambiguities, reflecting the different ways of translating the bare singular and bare plural determiners, and the preposition *"on"*. In this case, the transfer preferences determine that the best choices are to realise English bare plural as French definite plural, English bare singular as French definite singular, and *"on"* as an implicit temporal NP marker. Substituting these in, the preferred unpacked QLF is

```
elliptical_np(
  term(def_plur,
       C^[and,
           [vol1,C],
           form(temporal_np,
                term(def_sing,
                     E^[lundi1,E]))]))
```

producing the French surface output

les vols le lundi

12.2.5 Pre- and post-transfer

Ideally, we would like to say that unification-based rules and trainable transfer preferences constituted the whole transfer mechanism. In fact, we have found it necessary to bracket the unification-based transfer component between pre- and post-transfer phases. Each phase consists of a small set of rewriting rules, which are applied recursively to the QLF structure. It would in principle have been possible to express these as normal unification-based transfer rules, but efficiency considerations and lack of implementation time persuaded us to adopt the current solution.

The pre-transfer phase implements a simple treatment of reference resolution or coercion, which at present only deals with a few cases important in the ATIS domain. Most importantly, QLF constructs representing bare code expressions used as NPs are annotated with the type of object the code refers to. Code expression are frequent in ATIS, and the type of referent is always apparent from the code's syntactic structure. The extra information is necessary to obtain a good French translation: flight codes must be prefaced with *le vol* (e.g. *C O one three three* $\rightarrow$ *le vol C O cent trente-trois*) while other codes are translated literally.

The post-transfer phase reduces the transferred QLF to a canonical form; the only non-trivial aspect of this process concerns the treatment of nominal and verbal PP modifiers. In French, PP modifier sequences are subject to a strong ordering constraint:

locative PPs should normally be first and temporal PPs last, with other PPs in between. In the limited context of the ATIS domain, this requirement can be implemented fairly robustly with a half-dozen simple rules, and leads to a marked improvement in the quality of the translation.

It is worth noting that the structure of the pre- and post-transfer rule sets currently implemented is determined more by the idiosyncrasies of QLF formalism than by properties of particular transfer pairs. The greater part of the rules are used to handle the post-transfer canonicalization phase, and are language-pair independent.

12.2.6 Logical variables in QLF transfer

A minor point, but one which has caused us suprisingly many difficulties, is the question of how to handle logical form variables in QLF translation. In common with many systems built on top of Prolog, the CLE adopts the convention of representing logical form variables as Prolog variables. This can however lead to problems in QLF transfer; a piece of structure in the source side of a QLF transfer rule can unexpectedly unify with a logical variable, leading to an unwanted application of the rule. Problems of this kind become particularly acute when performing composition of QLF transfer rules (see Chapter 14).

The simplest solution to the problem is to replace all variables with unique constants during the pre-transfer phase of QLF transfer, restoring them again during post-transfer. Experimentation with the idea showed however that the resulting loss of expressiveness is a considerable hindrance to the transfer-rule writer. It is quite frequently desirable in complex transfer rules to map two distinct source-language variables into a single target-language variable; this is most easily accomplished if the three variables in question can all be unified.

Further examination of the practical requirements posed by transfer-rule writing revealed that a useful compromise was possible. Although it was sometimes necessary to unify two distinct source-side variables with each other, we found no instances where a good case could could be made for unifying a source-side variable with a non-variable. We have thus adopted the solution of "weak grounding" of logical variables during pre-transfer; each variable `V` is replaced with a term of the form `'VAR'(V1)`, where `'VAR'` is a constant not appearing elsewhere in transfer rules, and `V1` is a new variable. The effect is that QLF variables, as required, can be unified freely with each other, but not with anything else.

12.3 Robust transfer

12.3.1 Introduction

This section describes the robust translation architecture; as previously explained, its purpose is to provide a back-up mechanism to recover from situations where QLF-based processing has failed. The basic idea is to employ a second and much simpler transfer mechanism (word-to-word or WW transfer), which operates in parallel with QLF transfer. Word-to-word transfer is described in Section 12.3.2.

Both QLF transfer and WW transfer can in general be used on partial analysis results. This permits a bottom-up flow of control in the translation process. At various points in the source-side analysis process, the current best sequence of partial analyses is extracted, and the resulting package of information is sent over to the target-side transfer and generation process. The package of partial analyses is entered into a chart structure maintained by the taget-side process, and an attempt is made to translate each new item, using both the simple (WW) and sophisticated (QLF) translation methods.

The flow of control on the target side is similar to that on the source side; the process of translating edges in the translation chart can also be interrupted at an arbitrary point, and the current best spanning sequence of translated edges extracted. The intent is to achieve a translation architecture with the so-called "anytime" property. Processing can be halted at any time and an answer produced, the quality of the answer normally improving if additional processing time is allowed. The details of the chart-based translation architecture are the subject of Section 12.3.3.

12.3.2 Word-to-Word Transfer

This section describes the "word-to-word" (WW) transfer mechanism used to supplement the main QLF-based transfer component. The original intention was that WW-transfer would, as the name suggests, simply map each word in the source-language vocabulary into a target-language counterpart. A little experimentation convinced us that WW-transfer could be made slightly more expressive without abandoning the central goals of extreme simplicity and robustness. In fact, a more appropriate name would be "tagged-phrase-to-phrase transfer", but the original label has stuck.

A word-to-word transfer rule is a Prolog term of form

```
trule_ww([FromLang, ToLang], (FromPattern >= ToPattern]).
```

where

- `FromLang`, `ToLang` are language identifiers
- `FromPattern` is a list of one or more "word-to-word elements"
- `ToPattern` a list of zero or more "word-to-word elements"

A "word-to-word element" is either an atom representing a surface word, or a term of the form `Word/Category`, where `Word` is a surface word and `Category` is a major category symbol.

For example, the following are all valid word-to-word rules for the English → French language pair:

```
trule_ww([eng, fre], [to/p] >= [à]).
```

("translate 'to' to 'à' if it is a preposition")

```
trule_ww([eng, fre], from/p >= ['en partance de']).
```

("translate 'from' to 'en partance de' if it is a preposition")

```
trule_ww([eng, fre], [is/v, there] >= [y, 'a-t-il']).
```

("translate 'is there' to 'y a-t-il' if 'is' is a verb")

```
trule_ww([eng, fre], [how, about] >= ['q"en est il
de']).
```

("translate 'how about' to 'q'en est il de'")

A WW rule essentially declares that a tagged word-sequence matching the left-hand side may be translated into the word-sequence on the right-hand side. There are some slight complications deriving from the fact that we wish to be able to use the WW rules twice: first on raw output from the recognizer, and then later when lexical analysis has succeeded in giving input words plausible part-of-speech tags. In this way, we can conform more closely to the ideal of structuring translation as an "anytime" algorithm, which can be stopped at any instant and asked to deliver its best current guess at the answer.

Taking the two processing phases (raw recognizer output and lexical processing) in reverse order, we give the following definitions of what it means for a word-sequence to "match" the left-hand side of a WW rule:

The *lexical* transfer level operates on the result of linguistic analysis after lexical look-up and the first pruning phase. When this amount of processing has been performed, every word is associated with one or more lexical edges in the analysis chart (see Section 6.1), corresponding to the different possibilities offered by lexical and morphological processing.

A sequence of words

```
[W1,W2,...]
```

is defined to match the source side pattern

```
[X1/C1, X2/C2...]
```

at the lexical transfer level iff

1. The words themselves match, i.e. `X1 = W1`, `X2 = W2`...
2. The tags match, i.e. `W1` has an *unpruned* lexical entry with major category `C1`, `W2` has an unpruned lexical entry with major category `C2`...

The *surface* transfer level operates, as previously noted, on raw output from the speech recognizer. This motivates the following definition. A sequence of words

```
[W1,W2,...]
```

matches the source side pattern

```
[X1/C1, X2/C2...]
```

at the surface transfer level iff `[W1,W2,...]` is the word-sequence resulting from performing the following operations on `[X1,X2,...]`:

1. Lower-case each element of `[X1,X2,...]` if necessary.
2. If any element contains spaces, divide it up into its components.
3. Concatenate the results.

For example, at surface transfer level the list

```
[how, about, san, francisco]
```

matches

```
['how about', 'San Francisco'/np]
```

Note that at surface transfer level, the category information in the left-hand-side (source) pattern is irrelevant.

As can be seen, the WW transfer rule formalism has been designed to be very simple, so that rules can be written by relatively unqualified personnel. In practice, it turns out that the process of coding WW rules can be systematized to a high degree and turned into a mechanical form-filling task. The approach taken is closely related to that used in the `lexmake` tool (Section 7.1), and much of the code used is common to the two packages. The initial steps are the same: a suitable domain corpus is tagged by analysis up to the lexical level, and a set of contexts collected for each word/tag pair.

The results are written out as a file containing a "blank" set of WW rules, one for each word: the "blank" rule has the left-hand side pattern filled in with the word in question and the right-hand side pattern empty. A preceding comment lists the set of example contexts for the word. The transfer rule writer only needs to examine the contexts, decide on a translation of the current word appropriate in the given contexts, and fill in the right-hand side. If more than one translation is feasible, multiple copies of the rule may be created, or distinguishing context added to the left-hand side. Our experiences show that people with minimal knowledge of linguistics and no previous experience with the system are capable of writing WW rules at the rate of many hundred rules a day; for system experts, the figure can be in excess of 1500 rules/day.

12.3.3 Chart-based transfer

This section describes in more detail the "chart-based" architecture used by the robust transfer component. The overall purpose of this piece of the system is to coordinate and combine the two alternate transfer methods, QLF transfer and WW transfer, inside a single framework. The chart-based transfer mechanism is divided into two pieces, working within the source- and target-side processes respectively. We will start by examining the source-side half. As we will see, the basic functionality of its target-side counterpart is fairly similar.

Source side processing

On the source side, analysis of the input utterance progresses in a bottom-up fashion. As described in Section 1.2, prodessing goes through a number of stages, alternately adding edges to a chart structure and pruning them out. At various points, the current

most preferred sequence of edges is extracted and sent over to the target-side process. In the current version of the system, the source-side extraction operation is performed four times. We list the levels in turn:

Surface: Raw recognizer output. The top recognizer hypothesis is sent over unaltered to the target-side process, to be processed using "surface-level" WW rules (see Section 12.3.2).

Lexical: Called after the stages of tokenization, morphological and lexical processing, and lexical pruning (Section 6.2). The best sequence of chart edges is extracted and sent to the target-side process, to be translated by "lexical-level" WW rules (Section 12.3.2).

Phrasal: Called after the additional stages of phrasal parsing and phrasal pruning (Section 6.2). The best sequence of chart edges is extracted and sent to the target-side process, to be translated using QLF transfer and generation.

Full: Called when source-side processing has terminated, either normally or due to a time-out. The current best sequence of chart edges is extracted and sent to the target-side process, to be translated using QLF transfer and generation.

In view of the bottom-up nature of the parsing algorithm used for source-side analysis (Section 1.2), it would be easy to add further calls. An obvious candidate is an additional level intermediate between "Phrasal" and "Full", which would force extraction immediately after bottom-up analysis of NP phrases, but before full clausal parsing. We intend to investigate this idea during the next phase of the project.

The non-trivial part of the source-side robust processing method is the algorithm used to perform extraction of the best sequence of spanning edges from the chart. We have implemented two such algorithms, which we refer to as the "minimal" and the "stack-decoder" methods. We describe these in turn.

The "minimal" extraction method was implemented first. A formula assigns a score to each chart edge, and a simple dynamic programming algorithm is used to find the spanning sets of edges minimal with respect to the total score summed over all edges. If there are several such sets of edges, they are all returned, and the normal preference mechanism (Section 6.4.1) decides which is best. The formula currently used to score an edge is extremely simple. The edge receives a score of 1 if it represents a full (utterance-level) constituent in the grammar; otherwise it scores 2.

Somewhat to our surprise, performance of the minimal extraction method turns out to be reasonably good, reflecting the strong natural preference in favour of single coherent top-level analyses. There are in particular two common types of utterance on which it tends to score well: otherwise well-formed and correctly recognizer utterances to which the recognizer has incorrectly affixed one or two spurious extra words, and compound ("run-on") utterances consisting of two or more well-formed top-level utterances presented in rapid succession as a single utterance. None the less, the minimal extraction method suffers from obvious drawbacks, which lead us to believe that a more sophisticated approach is required in the long run:

- The preference ordering induced by the minimal extraction methods is entirely based on scores assigned to single edges. There is no way to take account of

interactions between edges: in particular, that an edge of type A is likely/unlikely to follow an edge of type B.

- The minimal extraction method makes no direct reference to the acoustic score delivered by the recognizer. This only comes into play as part of the normal preference method (applied at the end to adjudicate between co-minimal paths of edges), and cannot be used to justify choosing a longer path in favour of a shorter one, irrespective of their relative acoustic plausibility.

A preliminary version of a new and more sophisticated extraction method has been implemented, which is intended to address these issues. We summarize the method, which combines elements of the "stack decoder" architecture (Paul 1992) and the discriminant-based preference ideas of Section 6.4.

The basic design is modelled on the "stack-decoder". Processing begins at the start-node of the chart, and throughout the process a stack of at most N partial theories (for some suitable N) is maintained. A partial theory consists of a connected incomplete path of edges beginning at the start-node of the chart. The algorithm proceeds iteratively, each iteration consisting of the following steps:

1. Each partial theory (incomplete path) currently in the stack is extended with all possible next edges.

2. The partial theories are ranked using a discriminant-based preference method of the type described in Section 6.4.

3. The new stack consists of the N most preferred theories. The rest are discarded.

In order that partial theories should be mutually comparable, the transition between the extension and ranking steps is in fact a little more complex than stated above. All the theories included in the ranking step are chosen to be the same distance from the end-node of the chart, in terms of minimal separation by lexical edges. Theories closer to the end-point than this distance are saved, to be re-included in the stack at a later point.

The discriminants used to rank the partial theories in the stack are trained in the following way. First, a human judge simulates the stack-decoder process on a set of pre-processed chart data. A simple tool has been implemented which reduces the training process to menu-based selection of the best partial theory after each extension operation. The default choice, which is correct in at least 80% of all cases, is the longest partial theory of the set available. The result of the judging process is a set of sets of partial theories; each set contains at most one correct theory, the rest being incorrect. Sets not all of whose elements are incorrect are used to train discriminants in the way described in Section 6.4.

The heart of the process is the code which extracts "constraints" from partial theories; recall that discriminants summarize the reliability of constraints as indicators that a theory is respectively correct or incorrect. At present, we extract the following constraints from a partial theory T_i which is being compared with a set of alternate partial theories T:

1. Acoustic score constraint: the acoustic score for the acoustically worst edge in T_i.

2. Length constraint: the number of edges in T_i minus the number of edges in the shortest theory from T.

3. Syntactic N-gram constraints: 1- 2- and 3-grams of edge-categories. An edge-category is the major category symbol (`utterance`, `np`, `v` etc) of the syntactic constituent spanned by the edge.

4. Semantic N-gram constraints: 1- 2- and 3-grams of semantic edge-types. A semantic edge-type is a summary of the information in the QLF associated with the edge. We distinguish between different types of clause (imperative, Y-N question, WH-questions and declarative), elliptical phrasal utterances, and uninterpreted utterances.

We have so far only had time to carry out a few very sketchy experiments on the performance of the stack-decoder method, and it is not yet possible to say with any degree of confidence whether, as one would hope, it really does outperform the minimal method. Continuing this line of investigation is one of our top priorities in the next phase of the project.

Target side processing

The basic organization of robust processing on the target side is the same as that of the source-side processing described in Section 12.3.3. Recall that source-side processing at several points collects a current best sequence of analysis edges and passes them over to the target-side process. Each edge in the sequence is tagged with the indices of its start- and end-vertices in the source-side analysis chart. As the edges are received, they are entered into a target-side chart structure (the "target-side analysis chart" or "TSA chart"), whose vertices mirror those of the source-side analysis chart.

The target-side process then attempts to translate each edge from the TSA chart, entering the results in a second target-side chart structure (the "transfer chart"), whose vertices once again mirror those of both the original (source-side) analysis chart and the TSA chart. Translation of each edge is performed using a method appropriate to the package in which it was received, as described at the beginning of Section 12.3.3. "Surface" and "lexical" edges are translated using WW transfer; "phrasal" and "full" edges by QLF transfer. A lexical edge which cannot be translated by WW transfer is given a default translation as itself ("identity transfer"), to ensure that some spanning set of translated edges is always produced.

Just as on the source side, it is possible to interrupt target-side processing at an arbitrary point and extract the current best sequence of translation edges from the transfer chart; once again, the non-trivial problem is how to do this. We have implemented target-side versions of both the "minimal" and "stack-decoder" algorithms described in the previous section.

The "minimal" extraction method, as on the source side, assigns a score to each transfer edge and uses dynamic programming to find a spanning set of edges minimal

with respect to the total score. The scoring system is again very simple, and only takes account of the method used to perform translation, as follows:

- QLF transfer: score = 1.
- Lexical WW transfer: score = 2.
- Surface WW transfer: score = 3.
- Identity transfer: score = 4.

The basic effect is the same as in the source side version, namely to induce a strong preference towards finding the shortest sequence of edges translatable by the high-quality (QLF) translation method.

We have carried out some preliminary experiments using the stack decoder extraction method on the target side. A partial translation theory T_i, competing within a set of alternate theories T, receives the following constraints:

1. Length constraint: the number of edges in T_i minus the number of edges in the shortest theory from T.
2. Surface N-gram constraints: 1- 2- and 3-grams of translated surface words in T_i.
3. Translation method counts: number of edges in T_i translated by each possible method (QLF transfer, lexical WW transfer, surface WW transfer, identity transfer).

The use of the stack decoder method on the target side is again very much “work in progress”, and solid experimental results are as yet unavailable.

Chapter 13

Transfer Coverage

Ivan Bretan, Manny Rayner, Mats Wirén, Robert Eklund

The following is unchanged from the SLT-1 report chapter on Swedish transfer coverage. It still needs updating to reflect the coverage achieved under SLT-2. It also duplicates some material from other chapters in the current report.

In this chapter we describe the transfer component of the SLT system, starting out by describing the transfer formalism used. Section 13.2 then discusses the formalism's adequacy for the task with respect to its ability to deal with complex transfer phenomena, and its compositionality, simplicity and monotonicity properties. Section 13.3 how preference metrics can be used to solve transfer ambiguity problems, while the final sections of the chapter go into more detail on the different types of transfer rules used in SLT for translating ATIS-domain utterances.

13.1 The transfer formalism

The QLF transfer framework has basically been adopted unchanged from the Bilingual Conversation Interpreter (BCI) project (Alshawi *et al.*, 1991c), and will thus only be described briefly here. Unification-based QLF transfer is based on the notion of compositionally translating a QLF of the source language to a QLF of the target language, through matching QLF fragments against QLF pair patterns. By means of these patterns, transfer rules are specified declaratively using the following format:

```
trule(<Comment>,
      <QLF pattern 1>
      <Operator>
      <QLF pattern 2>).
```

The left hand side of this rule (`QLF pattern 1`) matches a fragment of the source language QLF and the right hand side the corresponding target QLF. The patterns can contain (possibly constrained) variables that match QLF fragments of arbitrary size, but no additional conditions can be associated with the rules. The resulting formalism

is therefore very simple, both from a semantic and syntactic point of view. The main idea behind keeping the formalism so simple is that only cross-linguistic data should be specified in QLF transfer. The particulars of how to form a QLF which corresponds to a grammatical sentence in a language is monolingual knowledge, and best left to the grammars, statistical preferences and lexica. The simplicity and compositionality of QLF transfer hinges on the fact that we can rely on the target grammar to filter out (refuse to generate from) certain QLFs.

The transfer algorithm can be seen as working with pairs consisting of a QLF fragment generated by the source language grammar in analysis mode, and a variable corresponding to the target QLF. The variable is (partially) instantiated to a target QLF fragment through unification of the pair with a transfer rule pattern. Besides instantiation, this matching produces a set of new pairs of source QLF sub-fragments and target QLF variables, which in their turn are matched against the rule set. When a QLF fragment does not match any pattern, it is decomposed into its functor and arguments, which are recursively translated. The QLFs output from transfer are to some extent ranked by the target language preference component (see Section 13.3), and finally submitted to semantic head-driven generation, using the target language grammar (Chapter 10).

The rest of this chapter will concentrate on reporting experiences from applying QLF-based transfer to the ATIS sentences. Since the BCI project demonstrated the strength and potential of this approach, we will here pay some attention to areas where the work with realistically collected dialogues has indicated that the framework needs to be extended.

13.2 Adequacy of the formalism

This fairly simple transfer rule formalism allows for succinct formulation of rules that deal with mappings between phrases that vary significantly in surface syntactic realization. The reason for this is the high level of abstraction in the QLF representation with respect to features essential for translation such as predicate-argument structure, mood, tense, aspect and modality, while keeping enough structural information to enable generation of a surface syntactic structure faithful to the original formulation.

In a true interlingua, all the work of determining what grammatical structures correspond to the interlingual representations is performed by monolingual knowledge sources. Besides the problems of interpreting natural language to a level as deep as the one required by an interlingua, there is the need for preservation of the syntactic and stylistic traits of the source expression. QLF as used as pivotal transfer representation in SLT is a "surfacial" semantic representation, in that it is not contextually resolved and quantifier scope has not been determined. On the other hand, it contains grammatical information that can be made use of in the translation process. Also, certain ambiguities that carry over from the source to the target language can be left unresolved. This "compromise" representation is less language-independent than a true interlingua.[1] Primarily, QLF is designed around the grammatical characteristics of a particular

[1]Note that QLFs are inherently language-dependent since all lexical semantic constants in the representations are entirely language-specific. That fact notwithstanding, there is still leeway enough in the actual

language, or group of languages. For instance, for the language pair English-Swedish we do not mark QLFs for *ablative* (source) or *allative* (goal), since these cases are not signaled by special grammatical features in either language, as opposed to Finnish where they are marked by affixing.

How does one then measure the quality of this compromise? The dimension of abstraction can basically be measured through the percentage of non-atomic transfer rules that are needed to accommodate differences in grammatical structures between English and Swedish (see Section 13.4.1 for statistics). The dimension of informational preservation is hard to measure in any other way than by manual evaluation of translations (see Section 10.4 of Agnäs *et al.*, 1994). On both these accounts, QLF-based transfer as implemented in SLT seems to score well. In addition, though, we would like the number of ambiguous mappings in the transfer rule set to be as low as possible, since extensive ambiguity would indicate the need for a deeper level of interpretation. This problem occurs in SLT mainly in connection with the translation of prepositions, and possible solutions will be detailed in Section 13.3.

13.2.1 Lexically triggered complex transfer

Given the fact that we have decided on a particular intermediate representation which will abstract over certain grammatical features it is important to assess what differences between source and target language expressions the formalism requires non-trivial (non-atomic) transfer rules to handle. We should expect a need for such rules where the grammatical structures of the two expressions are fundamentally different, and where this difference is localized to specific words or phrases. A typical example of such a difference is variation in argument structure. Such types of discrepancies between source and target language are well-known in the machine translation field, and besides difference due to purely idiomatic expressions, include the phenomena exemplified in Table 13.1.

Table 13.1: Examples of complex transfer phenomena

Complex transfer type	Example English-Swedish/German	
Argument switching	John **likes** Mary	Mary **gefällt** John
Head-switching	John **likes** swimming	John **schwimmt** gern
Object raising	John **wants** Mary to go	John **vill** att Mary skall åka
Passive to active	Insurance **is included**	Försäkring **ingår**
Verb to adjective	John **owes** Mary $20	John **är skyldig** Mary $20

For at least two of these phenomena, the current transfer rule formalism does not yet provide enough expressive power. Problems with handling object raising were noted already in the BCI report in association with rules such as the one translating between *"want someone to do something"* and *"vilja att någon ska göra någonting"*:[2]

use of the QLF format in a specific grammar to make it more or less language-independent.

[2] All QLFs and transfer rules exemplified in this chapter are simplified. Unresolved arguments and sometimes also the list of corresponding surface strings have been removed to improve readability.

```
trule(semi_lex(obj_raising,want_smn_to_do_sth-
                      'vilja_att_ngn_ska_göra_ngt'),
[want_DesireTo,A,tr(subj),
 [apply,tr(obj)^form(verb(to,no,no,to,y),
                     E,tr(body)),tr(obj)]]
==
[vill_att,A,tr(subj),
 [dcl,form(verb(pres,no,no,ska1,y),E,tr(body))]]).
```

This rule works fine in the English-to-Swedish direction, but not the opposite. This means that it is possible to unify *"John"* (`obj` above) with the subject of *"fly"* in *"Mary wants John to fly"* when translating that sentence into *"Mary vill att John ska flyga"* (lit. Mary wants that John should fly). However, translating from the Swedish sentence into English would require that the variable substitution be reversed. The solution suggested at that point was to provide a mechanism for non-deterministic generation of function applications through finding a QLF subexpression and replacing it with a λ-bound variable, such as `obj` above.

In addition, it seems as though the transfer framework also needs an extension to be able to cope with head switching. Head switching accounts for a particular type of discrepancy between source and target language syntax which can be observed in sentence pairs such as the ones in Table 13.2.

Table 13.2: Examples of head-switching

Head	Example English-Swedish/German/French	
like-schwimmen	John **likes** to swim	John **schwimmt** gern
want-simma	John **wants** to swim	John vill **simma**
come-être	John will probably **come**	Il **est** probable que John viendra

In these translations, the head word of the target sentence is not the translation of the head of the source. Thus, the arguments of the source verb need to be "moved" to the proper, arbitrarily embedded position in the source QLF. Consider the QLF pair:

```
[dcl,
 form(l([John,likes,to,swim]),
      verb(pres,no,no,no,y),A,B^[B,
       [like_LoveTo,A,
        term(l([John]),
             proper_name(tpc),C,
             D^[name_of,D,John]),
        form(l([to,swim]),
             verb(to,no,no,to,y),G,
             H^[H,[swim,G,v(C)]])]])]

[dcl,
 form(l([John,schwimmt,gern]),
```

```
verb(pres,no,no,no,y),A,B^[B,
 [gern,v(A)],
 [schwimmen,A,
  term(l([John]),
       proper_name(tpc),C,
       D^[name_of,D,John])]])]
```

Now, with the current transfer rule formalism there is no general way of specifying that the subject of *"like"* should correspond to the subject of *"schwimmen"*, without making the rule very specific to this example. In principle, there are two possible solutions to this problem. Either the transfer rule formalism is extended so that structural changes such as this can be expressed through a dedicated mechanism, or the QLFs produced by the grammar for this type of sentence are changed so that the current transfer rule formalism is sufficient to specify head-switching. The former solution seems to be the most promising to pursue, and would require a way to manipulate different significant parts of a QLF verb `form` without spelling out its structure in detail. In this case we would be interested in specifying that, in order to generate the German *"schwimmen"* VP, we need to translate the English verb predication for *"swim"* under the premise that the John `term` actually fills its subject slot, in practive "moving" the subject from *"like"* to *"swim"* (and this change of course needs to be reversible). Similar mechanisms have been used within comparable machine translation frameworks (e.g., in Kinoshita *et al.*, 1992).

13.2.2 Compositionality and simplicity

We can obtain a high degree of *compositionality* in transfer through relying on the principle of grammar filtering. In order not to compromise compositionality, transfer rule variable constraints were found to be useful, separating checking and changing of QLF elements. If we for instance would like to translate *"The flight arrives by five p m"* into *"Flygningen anländer innan sjutton noll noll"*, transfer needs to take two significant differences into account. Firstly, the specification of the time expression is different. In Swedish, 24-hour time is used rather than the *a m/p m* distinction. Secondly, when the NP argument of the preposition *"by"* is a time expression, we would like to restrict the Swedish translation to be the temporal preposition *"innan"* (although in general selection of target lexical item is a complex problem that needs special attention, see Section 13.3). Tentatively, we then end up with two rules. The first deals with time expressions:

```
trule(struct('a.m/p.m.'-'24hrs'),
      term(time(timeofday),B,
           A^[and,[hour_num,A,H],
                  [and,[minute_num,A,M],
                       [day_part,A,TimeOfDay]]])
      ==
      term(time(timeofday),B,
           A^[and,[hour_num,A,
```

```
                    @'24-hour-time'(TimeOfDay,H)],
                  [minute_num,A,M]])).
```

The second one translates *"by"* into *"innan"* in the context of `time` terms:

```
trule(struct(by_time-innan_tid),
      form(prep(by),A,
           C^
           [C,B,
            term(l(_),time(tr(time)),D,tr(restr))])
      ==
      form(prep(innan),A,
            C^
            [C,B,
             term(l(_),time(tr(time)),D,tr(restr))])).
```

Note however that these two rules partly refers to the same piece of QLF structure, namely the term with the *"time"* category. The consequence of this is that these two rules can never be used in the same translation of *"The flight arrives by five p m"*. In fact, since complex rules incapacitates compositional transfer, only the second rule above will trigger. In order to achieve the desired effect, we would have to specify a number of complex rules which combines the task of the two above ones, which of course violates the entire idea of compositional transfer.

An alternative solution would be to allow for *"by"* to translate into *"innan"* regardless of context. This may not seem so far-fetched if one recalls the design intentions behind QLF transfer, which is to minimize the duplication of source language knowledge. Transfer should be as compositional as possible and leave to the target grammar to decide on what constitutes a grammatical and preferred QLF. Still, there are reasons why such a solution is suboptimal. Even though maximizing compositionality through relying on grammar filtering is desirable, we also need to take efficiency into account. In certain circumstances, we know that a certain preposition translates in a specific way, and it is suboptimal to generate a set of possible translations which require totally different contexts to be valid. The more QLFs generated, the more work for the target grammar — ranking QLFs and attempting to generate from them (although packing the QLFs generated by transfer could alleviate this problem to a large extent).

Notwithstanding the compositionality argument against context-sensitive rules, the practical need for them seems to come up time and again. Thus, a natural extension to the transfer rule formalism has been to introduce a constrained variable specification, such as annotating transfer variables with a QLF pattern. Transfer rule variables can therefore be written either as `tr(<Id>)` or `tr(<Id>;<Pattern>)`.

In the second case, the rule will only be allowed to apply if the subexpression corresponding to the transfer variable matches `<Pattern>`, where "matching" would correspond to either unification or subsumption. The `<Pattern>` would constitute the variable constraint, the checking, whereas the `<Id>` variable enables compositional translation of the subexpression. The transfer rule handling the temporal use of *"by"* could then be reformulated as:

```
trule(struct(by_time-innan_tid),
```

```
form(l(_),prep(by),A,
     C^
     [C,B,tr(term;term(time(_),_,_))])
==
form(l(_),prep(innan),A,
      C^
      [C,B,tr(term;term(time(_),_,_))])).
```

Another issue needs to be considered in order to maximize compositionality of transfer, namely the need for grammars to generate QLFs compositionally. This is normally not a problem, but sometimes special design considerations give rise to QLFs that are not optimal from a compositional point of view. Most notably, the SLT grammars make use of the "big PP" construction (see Section 9.1.1), which is a normalized QLF representation of PPs that function as joint modifiers of a phrase. There are several reasons why this type of analysis seems to be motivated, but it does cause problems for transfer. For example, *"nonstop flights"* is naturally translated into *"flygningar utan mellanlandningar"* (flights without stopovers). However, a transfer rule achieving this mapping would not produce a QLF that the grammar could generate from if the source head noun were modified by a PP, as in *"nonstop flights to Boston"*, but would instead yield:

```
term(l([flygningar,utan,mellanlandningar,till,Boston]),
     q(_,bare,plur),_,
     B^[and,
        [and,[flygning_Flygresa,B],
             form(l([utan,mellanlandningar]),
                  prep(utan),_,
                  <"mellanlandningar">)],
        form(l([till,Boston]),
             prep(till),_,<"Boston">)])
```

Whereas the grammar would assume a QLF of the following, less compositional, format:

```
term(l([flygningar,utan,mellanlandningar,till,Boston]),
     q(_,bare,plur),B,
     C^[and,
        [flygning_Flygresa,C],
        form(l([utan,mellanlandningar,till,Boston]),
             conj(pp,implicit_and),_,
             X^[<"till boston">,
                <"utan mellanlandningar">])])
```

A solution suggested to this problem is to allow the transfer component to generate PPs compositionally, and having them restructured into "big PPs" in a post-processing phase. Another solution would be to include the compositional PP attachment rules in the grammar, but only allow them to function as generator rules.

Besides the problems noted here, the framework is highly compositional. In the BCI report, a number of combinations of complex transfer types and transfer contexts were tried out, revealing fourteen unwanted interactions where special rules were needed to handle the combination of phenomena, thus reducing compositionality. The transfer phenomena that were handled compositionally in the BCI are still handled compositionally in SLT, and in addition at least thirteen of the fourteen unwanted cases have been rendered spurious by the new QLF format, especially by the verb `form` "record" (see Section 9.4 of Agnäs *et al.*, 1994) which does away with operators for tense and aspect, which were one of the main sources of the unwanted transfer rules.

Simplicity follows the separation of monolingual and contrasting knowledge, as well as through the level of abstraction in the QLF format. For a language pair such as English-Swedish, the small amount of contrasting knowledge needed to cover the ATIS domain is very encouraging. And, as is indicated by the data presented in Section 13.4.1, the vast majority of transfer rules are atomic, and thus extremely simple to specify.

13.2.3 Monotonicity

The issue of monotonicity was discussed in some detail in the BCI report. In one respect, monotonicity was violated in the BCI since compositional transfer of a QLF expression would not be attempted if a complex transfer rule matched the expression. The main reason for enforcing this restriction is a concern for efficiency. And in practice, it does not seem to be major set-back. In certain cases in SLT, there are situations when both compositional and non-compositional transfer are possible, such as when translating sentences such as *"What are the `NPs`?"*, for which the following complex transfer rule was written:

```
trule(struct(what_are_x-'vad_finns_det_för_x'),
      [be,A,term(q(tpc,wh,_),B,E^[impersonal,E]),
       C^[eq,C,term(ref(def,the,plur,_),F,tr(restr))]]
      ==
      @det_finns(A,term(q(J,'vad_för',plur),
                 F,tr(restr)))).
```

It deals (non-compositionally) with for example the translation of *"What are the flights to Boston?"* into *"Vad finns det för flygningar till Boston?"*. But the compositional translation is still desirable, for instance in cases such as *"What are the arrival times in Washington D C?"* which preferably translates into *"Vilka är ankomsttiderna i Washington D C?"*, more or less a verbatim translation, which would be generated from the compositional transfer. However, it is not difficult to get around the restriction above in cases where it is necessary, such as this. It is simply a matter of specifying a second complex transfer rule which covers the same QLF fragment but instead maps to the corresponding QLF fragment that compositional transfer would generate. The above rule is therefore complemented with the following rule:

```
trule(struct(what_is_x-'vad/vilken_är_x'),
      [be,A,WhatTerm,
```

```
 B^[eq,B,tr(term)]]
==
[vara,A,@wh_ell(WhatTerm),
 B^[eq,B,tr(term)]]).
```

13.3 Dealing with transfer ambiguity

The major advantages of this framework, compositionality and simplicity, are as mentioned very dependent on the fact that the generate-and-filter approach works as intended. This is not only a question of blocking ungrammatical structures, but also of giving low priority to *unlikely* QLFs. For example, in ATIS the preposition *"in"* can translate to one of at least three Swedish prepositions depending on the context:

*"arriving **in** Boston"* ⇒ **till**
*"arriving **in** the morning"* ⇒ **på**
*"ground transportation **in** Denver"* ⇒ **i**

Rule-based approaches to transfer ambiguity would involve choice of the Swedish target word by inspecting the context of the source word in the English QLF. This is sometimes feasible, as in the above mentioned case of translating *"by"* when used with a time expression, although this approach would more generally require typed variables to be useful (in which case a rule could refer to all geographical places in order to translate *"arrive in"* + `Place` to *"anlända till"* + `Place'`). Besides requiring more rules, and thereby decreasing compositionality and simplicity, this approach duplicates a monolingual knowledge source, namely the semantic collocation preference triples (see Section 7.3.3). Let us therefore turn to preference-based approaches to transfer ambiguity, which make use of these triples to solve the transfer ambiguity problem. Preferences could potentially be applied at either the English or Swedish QLF stage to solve the problem. The relative merits of the two approaches are discussed below.

13.3.1 Preferences on English QLFs

Some cases of transfer ambiguity are due to a source language word having a number of genuinely different word senses, each with a different target language translation. For example in the ATIS domain the English verb *serve* has two distinct senses: `serve_Provide` (as in *"to serve a meal"*) and `serve_FlyTo` (as in *"flights serving Boston"*). In these cases it seems appropriate to have different word senses as separate entries in the source language lexicon, leading to QLFs for each word sense which preference metrics can then choose between. A separate transfer rule is thus written for each of the source language word senses:

```
trule(lex(simple),serve_Provide == servera_Något).
trule(lex(simple),serve_FlyTo == trafikera_Ställe).
```

This method could be extended to cases where there are no obviously distinct source language word senses, but transfer ambiguity occurs. Essentially a word would be split into word senses corresponding to each possible target word. Again, preferences at the

English side could be used to choose between these senses (in particular triple-based preferences would encode the necessary contextual information for this choice), and a separate transfer rule would be written for each "sense" of the preposition. However, this approach violates the principle that the SLT system is built out of stand-alone, modular components — English analysis has been fundamentally influenced by the translation task, and the target language in particular. Because of this, preferences at the source language side should only be used to solve transfer ambiguity problems when there are genuinely distinct word senses in the source language, as in the example for *"serve"* given above. This allows the English CLE to be developed purely through considerations of the source language. Problems similar to the preposition transfer ambiguity should be treated using preferences on the target language QLFs.

13.3.2 Preferences on Swedish QLFs

A promising approach to the transfer ambiguity problem allows the transfer stage to generate several alternative Swedish QLFs through non-deterministic transfer rules, and then applies preference metrics (in particular triples) to choose between these QLFs. For example the phrase *"arrive in Boston"* would result in three possible translations with corresponding triples:

"arrive in Boston" ⇒ *"anlända till Boston"* ⇒ `tr(anlända,till,Boston)`
"arrive in Boston" ⇒ *"anlända i Boston"* ⇒ `tr(anlända,i,Boston)`
"arrive in Boston" ⇒ *"anlända på Boston"* ⇒ `tr(anlända,på,Boston)`

In this case a high triple-score for `tr(anlända,till,Boston)` would result in the correct Swedish preposition being chosen.

The question is then how suitable scores for target-language (Swedish) triples can be derived. There are two obvious possibilities:

1. From a Swedish corpus.

In a symmetrical system which gives Swedish-English as well as English-Swedish translation, triple scores derived from a corpus for disambiguation of Swedish analysis will be required. These triple-scores could also be used for choice between Swedish QLFs generated during English-Swedish translation. In this way the transfer ambiguity problem is solved "for free".

It is anticipated that some care would be needed when applying this approach. The Swedish QLFs created by transfer from English QLFs may well be quite different in nature from those derived from analysis of Swedish text. For example it is unlikely that the triple `tr(anlända,på,Boston)` will be seen in a Swedish corpus, whereas the example above shows that it is likely to occur as the result of transfer from an English QLF. Preference metrics derived from a Swedish corpus would therefore give no score for triples such as this, so a suitable method of giving them a score would have to be found.

A training corpus could also be produced by allowing the SLT system to generate all translations for an English sentence non-deterministically, then to use human judgement to choose the best Swedish translation and with it the best Swedish QLF. In

this way a set of Swedish QLFs would be created for each sentence, with the best one tagged as such.

2. By hand-coding.

In the absence of a Swedish corpus for training triple-scores, they can be hand-coded. The process then involves: (1) identification of English words which have more than one possible Swedish translation; (2) identification of the context for which each Swedish target should be chosen; (3) encoding this choice by allocating scores to Swedish triples. This is the approach which was used in this work, and is described in the rest of this section. It is demonstrated that triples provide a useful form of abstraction which allows lexical choice through contextual considerations to be elegantly encoded.

13.3.3 Implementation of hand-coded triple-scores for transfer ambiguity

To demonstrate the use of hand-coded triple-scores, the method was implemented to solve the problems of translation of the prepositions *"in"* and *"on"* into Swedish. Having identified these words as sources of transfer ambiguity, solving the problem involved the following two stages:

1. Identification of the alternative Swedish translations and their context.

For this purpose a corpus of aligned English-Swedish sentences from ATIS, and help from a native Swedish speaker, were used. The following conclusions were made:

"In": In general this translates to *"i"* in Swedish, with the following exceptions:
"in the morning/afternoon/evening" ⇒ *"på"*
"arriving in Boston" ⇒ *"till"*

"On": In general this translates to *"på"*. The exceptions to this are
"information on . . ." ⇒ *"om"*
". . . on Delta/{etc.} Airlines" ⇒ *med*
(except for cases such as *"information on Delta Airlines"*, etc.)

2. Encoding of these rules by allocating triple-scores.

The scores below were chosen, the format being `ts(L,M,R,Score)` where `L`, `M`, and `R` are the left, middle and right elements of a triple, `Score` is the score for that triple. `cc_PartOfDay` denotes a collocation class, and will effectively unify with *"morning"*, *"afternoon"* or *"evening"*. Similarly `city_BigTown` will unify with any city name, and `airline_LineOfAircraft` will unify with any airline). Note that uninstantiated elements in the `ts` predicate (denoted by "_"), which will unify with any word sense, are used to capture generalisations.

Triple-scores for the translation of *"in"*:

```
ts(_,på,cc_PartOfDay,10)
ts(anlända_2p,till,city_BigTown,20)
ts(_,i,_,5)
```

For translation of *"on"*:

```
ts(_,på,_,2.5)
ts(_,med,airline_LineOfAircraft,10)
ts(information1,om,_,20)
```

13.3.4 Problems with hand-coded triples

There is a fundamental problem with the method outlined above. Encoding the triples by hand is laborious, and is only made possible by leaving triple elements uninstantiated in order to capture useful generalisations (without this technique every possible triple would have to be scored, a vast amount of work). However these uninstantiated triples can lead to conflicts. For example suppose that both *"in"* and *"on"* could translate to the targets *"på"* and *"i"*, but with different defaults:

"in" ⇒ *"i"* (default)
⇒ *"på"* (in some exceptional cases)

"on" ⇒ *"på"* (default)
⇒ *"i"* (in some exceptional cases)

Then writing default scores for the prepositions in the form `ts(_,prep,_,Score)` becomes impossible as the two defaults will conflict. Fortunately in reality *"on"* does not translate to *"i"* in the Swedish corpus, so the problem does not arise in this case. Problems such as this could be solved by effectively having a different set of triple-preferences for each of the English source words. However this again violates the principle of modularity, as Swedish preferences will have been fundamentally influenced by considerations of the English language. This problem suggests that use of a Swedish corpus to derive scores for *specific* triples, combined with the use of similarity measures to deduce scores for "missing" triples, would be a far more satisfactory approach. By avoiding making generalisations using uninstantiated triple elements the problem of "clashes" as described above should be avoided.

The SLT framework also uses structural preferences that penalize unlikely QLF structure, but currently only in analysis. It is quite possible to apply these preferences also on the QLFs generated by transfer, although it is not clear what the gain would be. So far, efficiency considerations have prevented the general application of preferences in transfer. Since QLFs are unpacked, we would have to generate each QLF before we could rank them. Another, more efficient, possibility is to use the preference triples locally during transfer, so that for instance the selected target prepositions are ranked without having to rank all possible target QLFs.

13.4 Rule types

Transfer rules can be divided into different classes. In the BCI report, this division was done on the basis on how *lexical* a particular rule is, which is a notion that will be used here as well. Thus, rules will be divided into the following groups:

- *Identity*: Rules whose left- and right-hand sides are identical atomic expressions.
- *Atomic lexical*: Rules whose left- and right-hand sides are distinct atomic expressions, normally corresponding to specific lexical items.
- *Non-atomic lexical*: Rules related to specific lexical items whose left- or right-hand sides are distinct non-atomic expressions.
- *Structural*: Rules whose left- and right-hand sides are distinct non-atomic expressions, neither of which is related to one distinct lexical item.

This division will be used in the overview of the transfer rules below. Note that the definitions of non-atomic lexical and structural rules may be slightly different from the BCI classification, where structural rules that translate a particular lexical constant in a particular way depending on the surrounding context were counted as lexical. Here, however, we will refer to such rules as structural. In fact, a large part of the structural rules can said to be "lexically triggered" in some sense. The structural rules constitute a small portion of the entire rule set, as can be seen in Tables 13.3 and 13.4), which show the type distribution of the transfer rules with and without macro-expansion (macros are described in Section 9.4.3 of Agnäs *al*, 1994).

Identity rules indicate that certain QLF atoms are language independent (or possibly specific to both languages), whereas lexical rules constitute the bulk of contrasting knowledge for a language pair such as English-Swedish. Non-atomic lexical rules translate a specific lexical item (possibly including complements), where the translation is not straight-forward for some reason (grammatical features or complement pattern could be different, a word can translate into a phrase, a phrase into a phrase, etc.). Structural rules are "interesting", in the sense that they indicate fundamental differences between equivalent expressions in the two languages which are not local to a lexical item.

Table 13.3: Distribution of unexpanded transfer rules over rule types

Rule type	Unexpanded	
Atomic lexical	614	71.5%
Identity	161	18.8%
Non-atomic lexical	52	6%
Structural (non-lexical)	26	3%
Other	6	0.7%
Total	859	100%

Table 13.4: Distribution of expanded transfer rules over rule types

Rule type	Expanded	
Atomic lexical	614	63%
Identity	161	16.5%
Non-atomic lexical	87	9%
Structural (non-lexical)	106	11%
Other	6	0.5%
Total	974	100%

13.4.1 Statistics on rule types

As can be seen from the Table 13.5, the transfer rules that can be used bidirectionally, i.e., both for English-to-Swedish and Swedish-to-English translation comprise the vast majority of the rules. Bidirectional rules are indicated by the == operator. The fact that there are half as many Swedish-to-English (=<) rules as there are English-to-Swedish (>=) is mainly a consequence of SLT being concerned with the latter direction only. When Swedish-to-English translation will be addressed for this domain, it is very likely that a significant number of new rules will have to be added when scaling up the Swedish grammar and lexicon in order to provide sufficient coverage for a realistic corpus.

Table 13.5: Reversibility of unexpanded transfer rules

Rule type	==	>=	=<
Atomic lexical	527	56	31
Identity	161	0	0
Non-atomic lexical	43	9	0
Structural (non-lexical)	14	12	0
Other	2	4	0
Total	747	81	31
Percentage	87%	9.5%	3.5%

13.5 Overview of the rules

In what follows, rules will be presented according to linguistic categories associated with them. For each category, one or more examples will be given illustrating the types of transfer rule that have been called for within the ATIS domain.

13.5.1 Identities

There are 161 identity rules, all of the format `atom == atom`. An example rule is

```
trule(lex(identity),temporal_np == temporal_np).
```

This rule expresses the fact that the `temporal_np` constant means the same thing in English QLFs as in Swedish. Identity rules are not expected to vary from domain to domain.

13.5.2 Proper names

There are 94 atomic rules for the translation of proper names, including names of cities, airports, and airlines, e.g:

```
trule(lex(simple), california_State ==
                   kalifornien_Delstat).
```

It is was necessary to have separate semantic constants for proper names and their abbreviations, otherwise odd translations could be the result within the context of the use of *mean*. For instance, if *"AA"* is an abbreviation of *"American Airlines"*, and these two names map to the same QLF constant in either the source or the target language, then there is the risk of generating *"Vad betyder American Airlines?"* ("What does American Airlines mean?") from *"What does AA stand for?"*.

There are 38 additional atomic rules representing names of days of week, months, etc.

13.5.3 Nouns

There are 160 atomic rules for the translation of atomic noun constants. Most of these seem to be relatively domain-dependent, and domain dependence has been exploited by eliminating alternative lexical mappings for some constants which are not relevant for the domain, or through ordering the lexical rules (properly trained collocation metrics would make this unnecessary). Take for example *"connection"*, as used in *"Which airlines have connections between Pittsburgh and Baltimore?"*, which in this domain normally translates into *"anknytning"*, *"anslutning"* or *"förbindelse"*. However, in the more general sense, as in a connection between two events, the proper translation is *"samband"*. And when used to indicate a personal relationship, *"kontakt"* (contact) is probably the most suitable word. Lexical transfer could be made trivial if the source analysis made a distinction which is fine-grained enough to produce different semantic constants for each different type of target word. This is obviously impossible without making the source grammar sensitive to the needs of the target grammar, which nullifies the idea of separating monolingual and contrasting knowledge, as discussed in Section 13.3.

Thus, we will have to be content with the granularity of lexical semantic analysis that has been judged to be appropriate for source language analysis purposes. In the case of *"connection"* and most other nouns in ATIS, this did not cause any problems, since we could rely on the restrictedness of the domain. Generally, however, when there

is a difference in semantic fields the same problem as when translating prepositions will arise, and the same solution applies (use of preference triples).

There are 10 unexpanded non-atomic rules which deals with the translation of specific nominals. Five of these deal with translating between compounds and non-compounds or vice versa, as in the following rule which translates *"round trip"* (which is a non-compound in the English ATIS lexicon) into *"tur och retur-resa"* (lit. there-and-back trip):

```
trule(semi_lex(complex,round_trip-'tur_och_retur-resa'),
      ['round_trip_TripWithReturn',B]
      ==
      [and,[tur_och_retur1,B],[resa1,B]]).
```

Some of these rules could be rendered unnecessary if the morphological component were more sophisticated. For instance, *"travel arrangement"* is translated into the "non-compound" *"researrangemang"*, which in reality is a compound as well, consisting of *"resa"* (travel) + *"arrangemang"* (arrangement). If it were treated as a compound in Swedish, translation would be compositional. Of the five remaining rules, only two are "interesting", since the other three are just there to overcome inconsistencies in the coding of the Swedish and English lexica. One of these rules maps the mass noun *"money"* into the (historically and morphologically) plural Swedish noun *"pengar"*. The other translates *"noon"* into *"(klockan) tolv"* (twelve o'clock).

13.5.4 Adjectives

There are 44 atomic rules dealing with adjective constants. Several of these are relatively general, such as *"great"*, *"different"*, or *"full"*. These words being general means that their translation is normally context-dependent. Again, we can rely on our ATIS corpus data, so that we with some confidence can claim that *"great"* means *"stor"* (big) rather than *"utmärkt"* (excellent). When these tricks no longer suffice, target QLF preference metrics will help out.

Translating the adjective *"interested"* used with a PP complement as in *"I am interested in a flight"* needs to take into account that the corresponding Swedish sentence *"Jag är intresserad av en flygning"* contains a passivized verb ("interested by"). The transfer rule needed to accommodate this structural gap would be relatively complicated, but was rendered unnecessary since *"intresserad"* was eventually classified as an adjective with a PP complement, just like in English, since this type of passive was not yet assigned proper QLFs by the Swedish grammar.

There is only one non-atomic lexical rule dealing with adjectives. The non-atomic lexical adjective rule maps *"least expensive"* into the non-periphrastic *"billigaste"* (which sounds more natural in Swedish). See also Section 13.5.14 below where structural changes related to adjectives are described.

13.5.5 Prepositions

There are 80 atomic lexical rules for prepositions, although many possible lexical mappings of prepositions have been left out in order to avoid combinatorial explosions of

target QLFs. In addition, we need preference triples for target language preposition collocations in order to select among the generated QLFs, and currently we only have such triples for the prepositions *"in"* and *"on"* (see Section 13.3). As an example of the ubiquity of prepositions, consider the following possible translations of *"for"*, of which only the first is used in the ATIS domain:

```
trule(lex(simple),for == för).   % a dinner for two
trule(lex(simple),for == mot).   % sail for England
trule(lex(simple),for == till).  % a meeting for next Monday
trule(lex(simple),for == åt).    % a ticket for John
trule(lex(simple),for == i).     % travel for a day
trule(lex(simple),for == på).    % the price for the ticket
trule(lex(simple),for == om).    % ask for help
trule(lex(simple),for == under). % it rained for months
trule(lex(simple),for == sedan). % waiting for several days past
```

One can of course question the value of such general rules, but the generality reflects the broad use of prepositions, and can only be resolved through information on what kind of collocations a preposition is normally used in or through resolution into semantic primitives so as to achieve transfer on a deeper level of interpretation.

13.5.6 Pronouns

There are seven atomic lexical rules for pronouns, and ten non-atomic lexical rules. Among the non-atomic, two deal with the translation of *"you"*, which depending on number can be translated into either *"du"* or *"ni"*:

```
trule(semi_lex(complex,you-du),
      ref(tr(a),you,sing,tr(b))
      ==
      ref(tr(a),du,sing,tr(b))).
trule(semi_lex(complex,you-ni),
      ref(tr(a),you,plur,tr(b))
      ==
      ref(tr(a),ni,plur,tr(b))).
```

Six of the non-atomic rules translate demonstratives, which are normally translated in one of two ways, either "literally" or "colloquially". The literal translation is a single-word definite pronoun, whereas the colloquial uses a composite demonstrative consisting of the determiner and a locative adverbial, *"här"* (here) or *"där"* (there):

```
trule(semi_lex(complex,those-de),
      ref(tr(pro),those,plur,_)
      ==
      ref(tr(pro),den,plur,_)).
trule(semi_lex(complex,those-'de_där'),
      ref(tr(pro),those,plur,_)
      ==
      ref(tr(pro),'den_där',plur,_)).
```

13.5.7 Adverbs

There are 14 lexical atomic rules for adverbial constants, including sentential, temporal and interrogative adverbs. Note however that some adjective constants also double as adverbs (e.g., *"one way"*, *"round trip"* and *"nonstop"*).

There are six lexical non-atomic rules for adverbial constants. Two of these are interesting since they map between English PP forms originating from `prep + NP` and Swedish PP forms derived from a single adverbial, as in:

```
trule(semi_lex(complex,from_where-'varifrån'),
      form(prep(from),B,
           C^[C,D,term(q(tpc,wh,_),G,
                       H^[place,H])])
      ==
      form(prep('varifrån'),B,
           C^[C,D,term(q(tpc,wh,_),G,
                       H^[place,H])])).
```

As can be seen from this rule, the two QLF fragments are very similar, and the only constant that differs is actually the preposition itself. In theory, a single lexical rule translating *"from"* into *"varifrån"* would suffice. However, `varifrån` is not a surface preposition, and only exists as a preposition in the QLF format. It is simply one part of the non-atomic analysis of the adverbial *"varifrån"*, and should only be generated in this specific context. A possibility would of course be to allow overgeneration, and let the target grammar fail on ungrammatical QLFs generated by a `from == varifrån` rule.

13.5.8 Determiners

There are 13 atomic lexical rules for determiners and eight non-atomic lexical rules, of which only two or three signal significant differences between the source and target language, for example the rules which translates the complex determiners *"all the"* and *"only the"* into simplex determiners, which corresponds to dropping the definite article.

13.5.9 Verbs

There are 144 atomic lexical transfer rules for verb constants, including constants representing multi-word phrases, such as verbs taking complex complements involving partitives or reflexives. An example is the mapping between *"ta om"* (lit. take over) and *"repeat"*:

```
trule(lex(simple),repeat_SayAgain =< 'ta_Om_Något').
```

The verbs *"be"* and *"have"* are also translated as lexical constants. As semantically vague verbs, these two words can of course translate into a multitude of target words. In order to avoid combinatorial explosions, these contextually sensitive translations are deferred to structural rules, which decreases compositionality, but increases efficiency.

An example of this is the translation of *"be"* into *"finnas"* (to exist), which is governed by a number of structural rules (see Section 13.5.11 below which discusses copula rules). There are only two non-atomic lexical rules for verbs, namely *"arrive"* and *"depart"*, which when used with a direct object, as in *"arrive Boston"*, need to be translated into a VP with post-modifying PP, *"anlända till Boston"*:

```
trule(semi_lex(complex,arrive_TurnUpAt_place-
                       anlända_till_ställe),
      [[arrive_TurnUpAt,D,tr(subj),tr(place)]] >=
      [form(prep(till),_,F^[F,v(D),tr(place)]),
       [anlända_2p,D,tr(subj)]]).
```

13.5.10 Mood, tense, aspect, and modality

There are two atomic rules translating logical constants used to signal mood feature values:

```
trule(lex(simple),sai_do >= no).
trule(lex(simple),emphatically_do >= no).
```

The first eliminates subject-aux inversion, which is not used in Swedish. The fact that a QLF represents a yes/no-question will be sufficient to generate the corresponding Swedish sentence through inversion of subject and verb. The rule should be bidirectional, but since `no` is used to signal several different feature values, the rule is currently restricted this way. Emphatic *"do"*, as in *"Do fly to Boston!"* does not have a corresponding auxiliary in Swedish, and needs to be expressed through prosodic features, which are currently not supported by the QLF format. A structural rule deals with transferring a *to*-infinitive into a bare infinitive, which is needed when translating a non-modal into a modal verb.

Another structural rule suppresses the progressive marker for finite verb phrases in order to provide for translating e.g, *"I am flying to Boston"* into *"Jag flyger till Boston"* (I fly to Boston). A more complex structural rule maps between `NP + progressive-VP` and `NP + relative clause` as in *"flights going to Boston"* $Rightarrow$ *"flygningar som går till Boston"* (flights that go to Boston). The rule is necessary since the Swedish lacks the possibility to mark the progressive aspect through the present participle in the same way as in English.

```
trule(struct(np_progressive_vp-np_relative_clause),
      [and,tr(head),form(verb(no,A,yes,M,D),
                         V,tr(restr))]
      >=
      [and,tr(head),
           [island,form(verb(pres,A,no,M,D),
                        V,tr(restr))]]).
```

Note that this rule interferes with the rules for modal verbs mentioned below, which also operate on the `verb` "records". This means that in for instance *"passengers wanting to go to Boston"* the verb *want* will not be translated into the modal verb *vill*. This

type of conflict could be eliminated through the constrained variable specifications discussed in Section 13.2.2.

There are 11 non-atomic rules to deal with specific modal verbs. Eight of them deal with uninteresting design differences between the treatment of modal verbs in Swedish and English. One handles the translation of the non-modal *"want"*, as in *"I want a ticket"*, into *"vill ha"*, which is a modal followed by *"have"* as main verb.

```
trule(semi_lex(modal_intro,
               want_WishFor-'vill_ha_Något'),
      form(l(_),
           verb(X,Y,Z,no,W),
           A,B^[B,[want_WishFor,
                   A,tr(term1),tr(term2)]],_)
  ==  form(l(_),
           verb(X,Y,@noprog(Z),vill1,W),
           A,B^[B,['ha_Något',
                   A,tr(term1),tr(term2)]],_)).
```

Another translates *"I would like a ticket"* (modal+main) into *"Jag skulle vilja ha en biljett"* (modal+modal+main). The last of these specific rules deals with translating *"I want to"* + VP into *"Jag vill"* + VP', which would not be a difficult translation were it not for the fact that *"want"* is classified as non-modal and *"vill"* as modal. This means that the subject of *"want"* needs to be "moved" to *"flyga"* (fly), i.e., a head-switching operation as described in Section 13.2.1. Since this type of complex transfer cannot be specified generally, a number of specific rules were coded to cover the cases that occurred in ATIS.

No special rules were needed to account for tense or diathesis (the active/passive distinction). However, it should be noted that future as expressed using *"will"*, as in *"I will fly tomorrow"*, should preferably be translated through the temporal verb *"komma att"*, as in *"Jag kommer att flyga imorgon"*, rather than using the direct translation of *"will"*, *"skola"*, which implies commitment.

13.5.11 Structural rules for copula

The largest group of structural rules, i.e., rules that act upon syntactic structures beyond a single lexical item, are the seven rules involving copula translations. Already in the BCI project, complex transfer rules accounted for four different translations of the copula from English to Swedish, accounting for the fact that *"be"* can translate into *"vara"* (literal translation), *"finnas"* (to exist), *"ligga"* (to be located), *"kosta"* (cost) or *"bli"* (to become) depending on the context. In addition to these alternative lexical mappings, there are many instances of idiomatic use of *"be"*, as in *"be in a hurry"*, which need to be translated by specific complex rules, and the same holds true in the opposite direction. In combination with a predicative adjective, *"be"* often needs to be translated specifically depending on the adjective. For instance, *"be right"* $Rightarrow$ *"ha rätt"*, *"be different"* $Rightarrow$ *"skilja sig"*, *"be ashamed"* $Rightarrow$ *"skämmas"*, *"be possible"* $Rightarrow$ *"gå"* (and again this occurs in the direction Swedish-to-English as well). Finally, a number of high-level syntactic

structural mappings are needed in order to translate constructions such as *"Your seat is 22A"* into *"Ni har sittplats 22A"* (You have seat 22A), where arguments are switched.

Of course, seven rules cannot possibly cover all of these uses of *"be"*, and as before the ATIS data have determined what rules actually needed to be included in SLT. These rules are classified as structural, since they take into account context in order to translate *"be"* properly. Theoretically, we could remove all rules which did not involve truly syntactic restructurings (such as the *"be"* $Rightarrow$ *"have"* transformation just mentioned) and replace them with lexical mappings. This would however lead to massive overgeneration of QLFs, an unduly inefficient solution which was not attempted. The following rule illustrates the context-sensitive translation of *"be"* into *"kosta"* (cost):

```
trule(struct(how_much_is_sth_Move-
             hur_mycket_kostar_ngt_Flytt),
      [be,A,tr(term),
       C^[eq,C,term(q(tpc,measure,mass),_,
                    J^form(how_much,D,E^[E,J]))]]
      >=
      [kosta_skjortan,A,
       tr(term),
       term(q(tpc,measure,mass),_,
            J^form(hur_mycket,D,E^[E,J]))]).
```

This rule will translate *"How much is a ticket to Boston?"* into *"Hur mycket kostar en biljett till Boston?"* (How much does a ticket to Boston cost?), and the contextual restriction is simply the QLF term representing the NP *"how much"*.

13.5.12 Structural rules for possessives

There are two structural rules dealing with possessive constructions. The first translates `NP1` *"of"* `NP2` into `genitive-NP2' NP1'`, e.g., *"the number of the flight"* into *"flygningens nummer"* (the flight's number), and the second produces an `NP1' prep NP2'` QLF. In many cases, this restructuring is unnecessary since the compositional translation would yield an acceptable result. However, the dedicated QLF possessive marker is not used in the Swedish grammar, since many different prepositions can act as possessive. It is therefore currently the task of transfer to map the genitive marker into the appropriate preposition. In the ATIS domain, it is feasible to restrict the target prepositions to *"på"* (on) and *"för"* (for), but in a more general framework more prepositions and extended preference metrics for ranking collocations would be needed.

```
trule(struct(the_x_of_y-'xet_prep_y'),
      term(tr(cat),
           C,D^[and,tr(pred),
                    form(genit(Poss),E,
                         F^[and,tr(pred),
                                [F,D,tr(term)]])])
```

```
>=
term(tr(cat),
     C,D^[and,tr(pred),
              form(@poss_prep(Poss),E,
                   F^[F,D,tr(term)])])).
```

13.5.13 Structural rules for temporal expressions

There are four structural rules dealing with temporal expressions. Three of them are related to the use of prepositions with temporal NPs. In one case a preposition needs to be added to the temporal NP, `NP Weekday` $Rightarrow$ `NP` *"på"* `Weekday'`. In another case it needs to be dropped, `NP` *"on"* `Date` $Rightarrow$ `NP' definite-Date'`. One rule translates times of day specified using the *a m/p m* distinction into 24-hour time, as mentioned in Section 13.2.2.

13.5.14 Structural rules for adjectives

There are three structural rules triggered by adjectives. Two of them translate nouns modified by adjectives into adjective-noun compounds, which in Swedish differ from the ordinary NP with an adjective both with respect to orthography, prosody and semantics. For instance, *"one way flight"* translates into *"enkel-flygning"* rather than *"enkel flygning"* (simple flight). The reason why two rules are needed is to accommodate other nominal modifiers which need to be conjoined with the noun QLF on a level higher than the adjective:

```
trule(struct(adj_n-'adj-n'),
      [and,tr(np),Adj]
      ==
      [and,@one_way_adj(Adj),tr(np)]).

trule(struct(n_pp_adj-'adj-n_pp'),
      [and,[and,tr(np),tr(pp)],Adj]
      ==
      [and,[and,@one_way_adj(Adj),tr(np)],tr(pp)]).
```

These rules are currently restricted to a small class of adjectives which trigger this compound formation.

13.5.15 Other rules

There are 19 lexical atomic rules for interjections, of which *"please"* and its translation *"tack"* also can act as a sentential adverbial.

Four non-atomic rules translate conjunctions of the form `conj(tr(cat),and)`.

Six structural rules are needed just to discard annotations of QLF structures that are not needed for the translation, such as the `l/1` term containing the list of surface strings:

```
trule(struct(l),l(_) == l(_)).
```

In addition to the above mentioned rules, seven structural rules of varying types remain.

Chapter 14

Transfer Composition

Manny Rayner, Ivan Bretan, Mats Wirén

This chapter describes a software reusability technique which we have successfully used to reduce the effort needed to construct sets of transfer rules for new language-pairs. Exploiting the clean declarative semantics of our transfer rule formalism, we show how simple methods borrowed from logic programming can be used to compose two sets of transfer rules for the language-pairs $L_1 \rightarrow L_2$ and $L_2 \rightarrow L_3$ into a new set for the pair $L_1 \rightarrow L_3$.

Automatically composed sets of rules are generally of lower quality than hand-coded sets, but can rapidly be improved by hand to an adequate level. We sketch the basic debug-and-test cycle used, and report initial results of experiments in which the existing Swedish $\rightarrow$ English and English $\rightarrow$ French transfer rule sets were composed into a set for Swedish $\rightarrow$ French. The resulting transfer rules have been used to construct a Swedish-to-French version of the main SLT system.

14.1 Introduction

There is a well-known argument in the machine-translation literature which goes something like this. Suppose that we want to build a multi-lingual MT system, i.e. a system which covers several languages, and which can translate from any one of them into any other. If our MT framework is transfer-based, and we have N languages, we will need to implement $N(N-1)$ sets of unidirectional transfer rules. Considering how much work is involved in implementing even one set of transfer rules, N need not be very large for a serious problem to arise. Even $N = 3$ will be enough to make one look for a way to simplify the task.

At this point in the story, the idea of an interlingua is often introduced. If we can translate from each language into the interlingua, and from the interlingua into each language, then we will only need to implement $2N$ sets of rules. As soon as N exceeds 3, we are winning. Unfortunately, and despite various claims to the contrary, we are still a long way from knowing how to build robust interlingua-based systems; indeed,

there are reasonable philosophical arguments for believing that such things may be impossible in principle. Right now, at any rate, transfer is our main alternative if we want to build even moderately large systems that perform useful tasks. So we are back where we started.

Rather than describe yet another angle on the interlingua approach, we will in this chapter explore a less common attempt to get around the obstacle. Suppose we have three languages, L_1, L_2 and L_3, and that we have implemented sets of transfer rules for the pairs $L_1 \rightarrow L_2$ and $L_2 \rightarrow L_3$. Then we already have a possible way of translating from L_1 to L_3: we translate into L_2 using the first set of rules, and then apply the second set to the result to get an output in L_3. Of course, this has also been tried before. The problem is that today's MT systems are so far from perfect that even one translation step gives output of dubious quality; the result of two or more successive steps is generally too poor to be useful.

It is here that we think we have something new to offer. In contrast to most of the non-trivial MT systems reported in the literature, the SLT transfer component has been designed to consist almost exclusively of declaratively specified information. More specifically, a version of the system for a given transfer pair and domain contains two types of knowledge: unification-based transfer rules, and numerical preference information. As explained in detail in Chapter 12, transfer proceeds in two phases. The rules define a set of possible transfer candidates, and the preferences then select the most plausible of them as the output translation.

Since the information in our system's transfer components consists of declaratively expressed rules, it is possible to compose off-line the sets of transfer rules for the $L_1 \rightarrow L_2$ and $L_2 \rightarrow L_3$ language-pairs, and produce a set of rules for the $L_1 \rightarrow L_3$ pair. This automatically composed set of rules will in practice be of considerably lower quality than either of the two original sets. However, our experiments to date indicate that it is possible to use the automatically composed rule-set as the initial point in a standard test-and-debug cycle, which can quickly improve it to a useful level of performance; the effort involved is very much less than that which would have been required to construct a good $L_1 \rightarrow L_3$ rule-set from scratch. Transfer rule composition can thus be viewed as a kind of software reusability technique.

The rest of the paper is structured as follows. Section 14.2 describes the transfer-rule composition operation. Section 14.3 describes the methodology used to improve the initial set of composed rules, and Section 14.4 the results of practical experiments carried out using the Swedish $\rightarrow$ English $\rightarrow$ French language-triple. Section 14.5 concludes.

14.2 Automatic transfer rule composition

14.2.1 Transfer composition as a program transformation

This section explains the theoretical basis of the transfer composition idea. Since our transfer rule formalism has a clean declarative semantics, it is possible, using techniques borrowed from logic programming, automatically to compose rules from the sets $L_1 \rightarrow L_2$ and $L_2 \rightarrow L_3$ to produce a set of rules for $L_1 \rightarrow L_3$. We will be-

gin with a series of increasingly less trivial examples, showing in abstract terms how transfer rules can be composed. These provide the background needed to motivate the algorithm currently used to perform transfer rule composition; the algorithm itself is described a little later. In the remainder of this section, we will use `l1`, `l2`, `l3` to denote the source, intermediate and target languages in our prototypical instances of transfer rule composition.

As described in Section 12.2, transfer rules compile into Horn clauses. We will use the notation defined there, writing

```
trule([l1,l2], LHS >= RHS).
```

for the transfer rule in the `l1`-to-`l2` language pair, whose left-hand (source) side is `LHS` and whose right-hand (target) side is `RHS`. We write a compiled transfer rule in the form

```
transfer([l1,l2], LHS1, RHS1) <- Body.
```

This is interpreted as meaning that the QLF `LHS1` in language `l1` can be transferred to the QLF `LHS2` in language `l2` if `Body` holds.

Our first example is chosen to be as simple as possible. We are given the transfer rules (1) and (2), compiling to (1a) and (2a):

```
(1)   trule([l1,l2], a >= b).
(1a)  transfer([l1,l2], a, b).

(2)   trule([l2,l3], b >= c).
(2a)  transfer([l2,l3], b, c).
```

We also have the basic composition principle, (C):

```
(C)  transfer([L1,L3],Q1,Q3) <-
       transfer([L1,L2],Q1,Q2),
       transfer([L2,L3],Q2,Q3).
```

which says that `Q1` in `l1` can be transferred to `Q3` in `l3` if it can be transferred through the intermediate expression `Q2` in `l2`. For technical reasons, we will also find it convenient to assume the converse composition principle C′:

```
(C') transfer([L1,L3],Q1,Q3) ->
       transfer([L1,L2],Q1,Q2),
       transfer([L2,L3],Q2,Q3).
```

which says that transfer from `l1` to `l3` can *only* be accomplished in this way.[1] Now successively resolving (1a) and (2a) with (C), we get (3a)

```
(3a)  transfer([l1,l3], a, c).
```

a compiled form of (3):

[1] We are of course aware that the converse composition principle is at best an approximation to linguistic truth.

```
(3)   trule([l1,l3], a >= c).
```

Thus we have formally proved that if it is possible to transfer the QLF expression `a` in language `l1` to `b` in language `l2` and `b` in language `l2` to `c` in language `l3`, it is also possible to transfer `a` in language `l1` to `c` in language `l3`.

Our second example is slightly more complex, and illustrates composition of recursive transfer rules. We start with transfer rules (4) compiling to (4a) and (5) compiling to (5a):

```
(4)   trule([l1,l2],
            p(tr(v1)) >= q(tr(v1)))
(4a)  transfer([l1,l2],p(X),q(X1)) <-
         transfer([l1,l2],X,X1).

(5)   trule([l2,l3],
            q(tr(v1)) >= r(tr(v1)).
(5a)  transfer([l2,l3],q(Y),r(Y1)) <-
         transfer([l2,l3],Y,Y1).
```

Resolving C with (4a), we obtain

```
transfer([l1,L3],p(X),Q3) <-
   transfer([l1,l2],X,X1),
   transfer([l2,L3],q(X1),Q3).
```

and then resolving again with (5a) we get

```
transfer([l1,l3],p(X),r(Y1)) <-
   transfer([l1,l2],X,X1),
   transfer([l2,l3],X1,Y1).
```

We can now use C′ to replace the body of this last expression, obtaining

```
transfer([l1,l3],p(X),r(Y1)) <-
   transfer([l1,l3],X,Y1).
```

which as can be seen is a compiled form of

```
trule([l1,l3], p(tr(v1)) >= r(tr(v1))).
```

The two examples so far were both of the same form: schematically, we composed an `l1`-to-`l2` rule and an `l2`-to-`l3` rule, such that the right-hand side of the first rule was identical to the left-hand side of the second rule. It is unsurprising that such rules can be combined. What is less obvious is that rules can sometimes also be composed when the match in the intermediate (`l2`) language is not exact. The following is a minimal example.

We start with the `l1`-to-`l2` rule (6) compiling to (6a) and the `l2`-to-`l3` rule (7) compiling to (7a): for reasons that will shortly become apparent, we also need a second `l1`-to-`l2` rule, (8), with compiled version (8a):

```
(6)   trule([l1,l2],
            p(tr(v1)) >= q(tr(v1))).
(6a)  transfer([l1,l2], p(X), q(Y)) <-
        transfer([l1,l2], X, Y).

(7)   trule([l2,l3], q(b) >= r).
(7a)  transfer([l2,l3], q(b), r).

(8)   trule([l1,l2], a >= b)
(8a)  transfer([l1,l2], a, b).
```

We start by resolving the composition rule, C, with (6a), to get

```
transfer([l1,l3], p(X), Q3) <-
   transfer([l1,l2], X, Y),
   transfer([L2,L3], q(Y), Q3).
```

The resolving again with (7a), the result is

```
transfer([l1,l3], p(X), r) <-
   transfer([l1,l2], X, b).
```

Intuitively, what we have done so far is to translate the right-hand side of (6), i.e. `q(tr(v1))`, using (7), to get `r`. The condition left in the body shows that this is only possible if `tr(v1)` translates into the constant `b`.

The current result is not a well-formed `l1`-to-`l3` transfer rule, since it makes reference to transfer between `l1` and `l2`. The condition can however be removed by a further resolution with (11a), producing the final result

```
transfer([l1,l3], p(a), r).
```

which is the compiled form of the `l1`-to-`l3` rule

```
trule([l1,l3], p(a) >= r).
```

In effect, the final resolution step uses (8) to perform a backward (`l2`-to-`l1`) translation of `b`. Viewed in terms of transfer rules, we previously had the condition on the composed rule that `tr(v1)` translated into the `l2` constant `b`. The backward translation step showed that one way to satisfy this condition was to restrict the left-hand side (`l1`) occurrence of `tr(v1)` by forcing it to be specifically `a`.

14.2.2 Procedural realisation of transfer-rule composition

We will now describe our concrete algorithm for composition of transfer rules. We stress that this algorithm is not complete; there are many (in general, an infinite number) of valid `l1`-to-`l3` transfer rules which it will not discover. This appears to be inevitable, however, since the problem of generating a complete set of composed transfer rules is in the worst case highly intractable. (In fact, we strongly suspect that it is undecidable). Our experimental findings, described in more detail in the next section, suggest though that the algorithm has considerable practical utility. By creating

a manageably small set of composed rules which contains all the intuitively straightforward compositions, it greatly diminishes the effort involved in creating a new set of l1-to-l3 rules.

As usual, we refer to our three languages as l1, l2 and l3, and compose in the order l1 → l2 → l3. The basic idea is to follow the procedure sketched in the examples from the previous sub-section. We start with two language-pairs; for each language-pair, we take each rule in turn from that language-pair, and successively attempt to compose it with all the rules from the *other* language-pair. Schematically, there are thus two cases, which we refer to as "forward" and "backward" composition. In the "forward" case, we take an l1-to-l2 rule

```
trule([l1,l2], LHS >= RHS)
```

and attempt to use l2-to-l3 rules to translate RHS into the l3 expression RHS′; this may involve adding restrictions to LHS to produce LHS′. The final rule will be

```
trule([l1,l3], LHS' >= RHS')
```

The "backward" case is similar; we start with an l2-to-l3 rule

```
trule([l2,l3], LHS >= RHS)
```

and use l1-to-l2 rules (in reverse) to translate LHS into the l1 expression LHS′; this may involve adding restrictions to RHS to produce RHS′. The final rule is again

```
trule([l1,l3], LHS' >= RHS')
```

Since the two cases are symmetrical, we can without loss of generality restrict ourselves to the forward one. In more detail, we go through the following steps for each l1-to-l2 rule

```
trule([l1,l2], LHS >= RHS)
```

Note that we use the *uncompiled* version of the rule, i.e. the one in which transfer variables are left unchanged.

1. Translate RHS into l3 using l2-to-l3 rules, giving the result RHS′
2. Transfer is carried out in such a way that a transfer variable tr(Id,Pat) may be unified with the term T iff Pat unifies with T. Call the list of all Id/T pairs produced in this way the "unification-pair list".
3. For each Id/T pair in the unification-pair list, translate the l2 expression T into an l1 expression T′, using the l1-to-l2 rules in the inverse direction. Perform this step non-deterministically, to create a set of possible "inverse unification-pair lists", each of which associates all the Id with their respective values of T′.
4. Use each inverse unification-pair list to restrict the transfer variables in LHS. Thus if tr(Id,Pat) is a transfer variable in LHS, Id/T′ is an entry in the inverse unification-pair list, and Pat unifies with T′, replace tr(Id,Pat) with tr(Id,T′). The result of performing all these replacements is LHS′.

5. The final composed transrule is

```
trule([l1,l3], LHS' >= RHS')
```

14.2.3 Composing transfer preferences

As explained in Section 12.2.1, the transfer preference model contains two components: a *channel component*, expressing the *a priori* plausibility of the rules used to perform a transfer, and a *target language component* expressing the *a priori* plausibility of the result.

In practice, the channel component is calculated by assigning a numerical score to each transfer rule, and summing the score over the multi-set of all rules used in performing a given transfer. The target language component is calculated from structural properties of the QLF produced; the details of how this is done are not relevant to the present discussion. The final preference score is a weighted sum of the channel score and the target language score.

Our initial approach to composition of transfer preferences is very simple. Let us assume as usual that we are composing `l1`-to-`l2` and `l2`-to-`l3` to create `l1`-to-`l3`. The target language component for the composed rule-set is simply the target-language component from the `l2`-to-`l3` transfer set. The intuition is that since the target-language component estimates the intrinsic plausibility of the resulting `l3` QLF, it should make no difference whether this result was produced by transfer from `l1` rather than `l2`.

The channel model score assigned to each composed rule is also calculated straightforwardly as the sum of the channel model scores for all of the rules used by the algorithm in the previous section to form the composition. Finally, the relative weights assigned to the target and channel components are (somewhat arbitrarily) defined to be the same as those for the `l2`-to-`l3` transfer model. Although the transfer preference composition method as currently defined is clearly no more than a coarse approximation, it does appear to be good enough to function as a reasonable starting point, and in practice preserves most of the strong preferences induced from the original transfer pairs. Problems pertaining to composed transfer preferences are discussed at greater length in Section 14.4.3.

14.3 Improving automatically composed rule-sets

Our basic system development methodology makes use of rationally constructed, balanced domain corpora to focus the effort on frequently occurring problems (cf Chapter 8). The idea is to start with a large source-language corpus, and automatically identify commonly-occurring syntactic patterns. A smaller, "representative" corpus can then be extracted, in which each syntactic pattern is represented by one instance; each instance is paired with a number showing how many words from the original corpus it represents.

The key advantage offered by the representative corpus idea is that it simplifies the task of locating problems which have a large impact on coverage; experience shows

that if a particular sentence fails to process correctly, there is a substantial probability that all or most of the other sentences conforming to its syntactic pattern will fail in the same way. Thus the size of the syntactic class represented by each representative corpus sentence provides a natural way of prioritizing problems.

The development cycle for improving a set of automatically composed transfer rules is the same as for other phases of the system. After the latest round of improvements has been added, the representative corpus is run, and the results evaluated by a bilingual judge. The priority order implicit in the corpus determines the next set of problems to be attacked, and the cycle repeats.

In the context of automatic transfer composition, three specific problems may arise: overgeneration of composed rules, problems with composed preferences, and undergeneration of composed rules due to lack of coverage in the hand-coded rules. The rest of this section briefly describes these generic problems at an abstract level. In the next section, we present practical examples taken from our experiences in the Swedish $\rightarrow$ English $\rightarrow$ French language-triple.

14.3.1 Overgeneration of composed rules

Although the transfer composition algorithm cannot in a sense find $L_1 \rightarrow L_3$ rules that are actually *invalid*, it can certainly create ones that are *irrelevant*. In practice, the main consequence of overgeneration in transfer rule composition is increased processing time due to spurious non-determinism.

If the composed $L_1 \rightarrow L_3$ rule set is found to contain undesirable rules, we allow the developer to supply declarations blocking the combinations that gave rise to them. (Each rule is tagged with a history identifier which makes this a simple operation). The format of the declarations allows blocking of classes of rule combinations, for example inhibiting composition of a given rule from $L_1 \rightarrow L_2$ with all but one of a set of potentially composable rules from $L_2 \rightarrow L_3$.

14.3.2 Composed preferences

As described in section 14.2.3, the initial set of composed preferences is only a rough approximation, and requires considerable adjustment to achieve high performance. The issues involved are described in more detail in section 14.4. More generally, if L_1 and L_3 have a lexical granularity in common which is not matched by that of L_2, we risk ending up with too general lexical transfer rules as the result of rule composition.

14.3.3 Lack of coverage in hand-coded rules

Holes in the $L_1 \rightarrow L_3$ set result either from blocked compositions (as described above) or from cases not covered by the $L_1 \rightarrow L_2$ or $L_2 \rightarrow L_3$ transfer rules. When such holes are found, the developer can plug them by writing her own rules, adding them to the automatically composed set. The hand-written and automatically composed portions of the new rule set are kept in separate files, so that the composed portion can be recompiled without affecting the hand-coded additions.

14.4 Swedish → English → French: a case study

This section describes our experiences in constructing a Swedish-to-French transfer module using the methods outlined above. The current prototype works in the same ATIS domain as the other versions of the system, using a vocabulary of about 1 500 source-language stem entries, and has been integrated into the main SLT system to allow translation of spoken Swedish into spoken French.

14.4.1 Experimental results

As described in the preceding section, most of our development work has been carried out using rationally constructed Swedish representative corpora. The main corpus we have used contains 654 sentences, representing between them a total of 13 990 words from the original Swedish corpus. We normally count coverage in terms of "weighted scores" on this corpus, multiplying the result on each utterance by the number of words it represents. On this metric, we currently obtain French translations for 87% of the sentences; when evaluated by a bilingual judge, 72% were found to be of adequate quality, and the remaining 14% were deficient in various ways. 13% of the sentences produced no translation. Only two of the cases where no translation was generated were caused by the system exceeding its time-out threshold, here set to 90 seconds. The remainder of the section describes these results in more detail.

14.4.2 Translation failures

When no translation at all is generated, this is normally due to the incompleteness of the two underlying transfer rule sets. This is especially apparent for $L_2 \rightarrow L_3$, in this case English to French. This transfer module performs relatively well on the standard collected English corpus data used for testing, but much worse on the type of English generated by the Swedish-to-English translation module. The degradation in transfer quality is only to be expected, since spontaneously generated English and English generated by translating Swedish differ in many important respects. This is at least true in a transfer framework, where generation of target language utterances is not independent of the syntactic realisation of the source language utterance (as perhaps could be the case in an interlingua-based system).

14.4.3 Incorrect translations

Table 14.1 summarizes the main causes of error in the utterances from the Swedish representative corpus which failed to produce adequate French translations. The column headed 'Freq' gives the number of sentences affected (frequency-weighted) for each category.

Correctness of translations is of course not a binary notion; we can distinguish several different types of reasons why translations fail to be satisfactory. In Agnäs *et al.* (1994) four different quality dimensions were used to evaluate the SLT system:

Type of problem	Freq	
Transfer preferences	1162	40%
Missing transfer coverage	889	30%
Swedish analysis preferences	228	7%
French grammar	180	6%
Other/unclear	476	16%
TOTAL	2935	100%

Table 14.1: Inadequate translations in Swe → Fre tests

- Degree of meaning preservation
- Degree of preservation of stylistic information
- Grammaticality
- Naturalness

In the above figures on correctness, we have mostly been considering the three first dimensions.

From the above figures we can see that there are two major problem sources when using the composed transfer rule set: preferences and missing transfer coverage. In addition, due to overgeneration of composed rules, problems with efficiency needed to be addressed through blocking declarations.

Dealing with overgeneration

In the current version of the system the Swedish-to-English transfer rule set contains 1 454 rules and the English-to-French rule set 1 281 rules (both sets counted after macro-expansion). The automatically composed Swedish-to-French transfer rule set contains 4 525 rules. As can be seen, it is considerably larger than either of the two hand-coded rule sets.

This disparity in size appears to be the main reason for the appreciably slower processing times delivered by the composed Swedish-to-French rule set, compared for example with the hand-coded English-to-French rules. Using the current version of the system and the English and Swedish representative corpora described above, the average processing time for the English-to-French transfer and generation phase is 5.3 seconds/utterance running Quintus 3.2 on an UltraSparc 1; 9.5% of the utterances (frequency-weighted) took more than 10 seconds. We ran a corresponding test on the the first 140 sentences of the Swedish-to-French representative corpus, representing 7780 words of the original corpus. This produced 14 time-outs, using a time-out threshold of 90 seconds; over the remaining sentences, we recorded an average processing time for the transfer and generation phase of 16.2 seconds/utterance, with 40.0% of the utterances (again frequency-weighted) taking more than 10 seconds. These figures are clearly not satisfactory.

Examination of the composed rule-set revealed that a substantial proportion of the rules were created by forward or backward composition (cf Section 14.2.2) with a small set of "problem rules". A typical offender is the Swedish-to-English rule which results in English day-of-week/time-of-day expressions like "Monday morning". These phrases are a specific type of compound nominal; English compound nominals can be translated into French in many ways, most often as noun/PP combinations. Consequently, forward composition from the Swedish-to-English rule results in a large set of Swedish-to-French rules. However, the only acceptable way to translate these particular compound nominals is as French compound nominals ("Monday morning" → "lundi matin" etc). The forward composition with the appropriate English-to-French rule is correct, and all the other compositions are irrelevant.

The above example illustrates how addition of a small set of blocking declarations (cf Section 14.3.1) makes it possible to effect a substantial reduction in the size of the composed rule-set. In the current version of the system, 53 declarations between them block creation of 2 126 out of the 4 525 composed Swedish-to-French rules. Suitable declarations were found by inspecting utterances for which transfer was clearly slow; the total time required to write them was on the order of 2–3 days. Rule-filtering using the blocking declarations reduced the average frequency-weighted processing time for transfer and generation from 16.2 seconds/utterance to 4.6 seconds/utterance, with the proportion taking over 10 seconds falling from 40% to 8%. The new figures are comparable with those for English-to-French quoted above.

Dealing with preference problems

The automatically composed transfer rule preferences are sometimes insufficient to select the correct lexical target constant. The problems which arise typically concern incorrect lexical choice when translating prepositions, articles and compound nominals. This section describes a typical case in detail.

Consider the translation of the Swedish preposition "på". In the Swedish-to-English transfer rule set the following lexical transfer rules exist: *på* → *on*, with channel score 10, and *på* → temporal-NP-marker, with channel score –5. The relatively high transfer rule score of 10 reflects the fact that the "default" translation of "på" should be "on".

Then, between English and French there are several different lexical transfer rules translating "on" into various prepositions, but none ranking higher than any other, not even the expected "default" translation "sur"; in the ATIS domain, there is in fact no strong default translation of "on". Thus for instance "on Delta" becomes "avec Delta", "on Monday" becomes "le lundi" (with the preposition being translated into an implicit temporal NP marker), and so on. Consequently, the Swedish → French transfer rules *på* → *avec* and *på* → *sur* both receive the positive channel score of 10.

Both *på* → *on* and *on* → *avec* are very reasonable transfer rules, and it is correct to assign a positive score to the first and an approximately zero score to the second. However, the composition *på* → *avec* is unlikely, especially in this domain. The problem is that the English-to-French rule *on* → *avec* is mainly appropriate in contexts where the object of "on" is an airline, which will frequently be the case in ATIS. However, the object of the Swedish preposition "på" is hardly ever an object of this type, and so the two transfer rules are in practice rarely capable of combination. A positive score for

the composed rule is thus incorrect.

The above example shows that straight-forward summing of preferences may sometimes lead to awkward results; when similar cases are discovered during system testing, the preference scores need to be corrected by hand. We hope that the new methods we are developing for semi-automatic acquisition of transfer preferences (cf Section 12.2.3) will help us attack these problems during the next phase of the project.

Dealing with lack of coverage in hand-coded rules

Just as the difference between "natural English" and "English generated through translation from Swedish" sometimes caused complete failure when using the composed rules for translation between Swedish and French, this difference can also lead to *incorrect* translations. Consider for example the Swedish construction "Vad kostar *X*?", having a literal English counter-part in "What does *X* cost?" which occurs seldom in the English training corpus used. Because of this, in the original composed set of Swedish-to-French rules, no correct rule existed for translating "What does *X* cost?" into "Combien coûte *X*?". This in turn led to the quite common construction "Vad kostar *X*?" being compositionally and erroneously translated into "Que coûte *X*?". In the figures in Table 14.1, however, a version of the system was used where the relevant rule had been added to the English-to-French rule set ("What" → "Combien").

14.5 Conclusions and further directions

To summarize, our basic idea is to abandon the quixotic search for an interlingua, and instead pursue the more modest goal of efficiently re-using existing linguistic resources. To this end, we have described an algorithm for composition of transfer rules. We have demonstrated that it can be used to automatically compose non-trivial sets of transfer rules containing on the order of thousands of rules, and shown that by small adjustments the performance can be improved to a level only slightly inferior to that of a corresponding set of hand-coded rules. Our experience is that the amount of work involved in using these methods is only a fraction of that needed to develop similar rules from scratch.

In the short-term, we intend to continue to work on hand-improvement of the automatically composed rule-sets; our aim here will be to ascertain how much effort is required to close the gap in performance between the automatically composed and the hand-coded rule-sets. We will also test our methods on two other language triples, English → French → Spanish and French → English → Swedish. Preliminary results on the first of these already look promising.

On the basis of the experience gained from these experiments, we will also try to improve the underlying methodology. In particular, it is important to obtain automatically generated rules that are more transparent. Our current plan is to develop methods for formatting trace information, so that it becomes easier to see how a particular composed rule was generated. In principle, such trace information is already available, but right now it is packaged in a way which makes it difficult to read and understand.

One option for long-term work would be to systematize the use of transfer rule composition in a multi-lingual MT framework. The rough picture we have in mind is to concentrate on a small number of "canonical" language pairs. What constitutes a canonical pair could either be determined from pragmatic reasons (in the European Union French–English and German–English would of course be prime candidates), but also from typological facts. Languages could be clustered according to linguistic similarity and only *one* representative language A_1 selected from language family A for manual translation into a single language B_1 from family B; then the $A_1 \rightarrow B_1$ rules could be composed with the (hopefully easy-to-construct) rule-sets obtaining within the language-families A and B. In this way, rules would be derived automatically between $A_2, \ldots A_n$ and $B_2, \ldots B_n$. If this programme of research proves viable, it could have interesting implications for the field as a whole.

Chapter 15

Database Interface

Ralph Becket, Ian Lewin, David Milward and Manny Rayner

15.1 Overview of Database Processing

In SLT database interface applications, database processing is the last stage in the translation pipeline. Its purpose is to take a logical representation of the speaker's request (expressed in a Target Reasoning Language, TRL) and transform it into an (approximately) equivalent TRL sentence which can be "executed" to obtain the answer to the query.

The utterance from the speaker is translated into a logical formula conveying the meaning of what was said in order to support reasoning and manipulation of it within a well understood framework. As might be expected, there is still a world of difference between the way something is going to be described in human terms even in a logical formalism and the terms which a database query would use (the final, translated form is essentially a program for extracting the requested data from the database).

The approach taken in the ATIS SLT project is called Abductive Equivalential Translation, about which we will say more later. The key point is that it is logically clean, so it is possible to make sensible claims about what the system can do and how it can be used to attack new problems; it has an efficient implementation, so query response times are fast enough to be useful; and it is ideally suited to handling the kind of context-dependent constructions prevalent in natural language (consider the difference between, say, any flight, a morning flight and an early morning flight).

Moreover, since it will not always be possible to give a perfect answer, AET can identify situations where assumptions have to be made (e.g. in its understanding of what constitutes early morning) and report them to the user. Similarly, the system can be used to identify errors such as references to flight numbers not listed in the database. All of this is handled in a straightforward way without the need for special case solutions or auxiliary mechanisms.

So, essentially, the database processing part of the system is responsible for translating natural-language style sentences in the target reasoning language (TRL) into

executable database queries, also in TRL form.

In order for a TRL to be executable it must consist entirely of executable predicates (i.e. ones for which there exist Prolog definitions). Such predicates include comparison tests, conversion operations, ordering predicates (e.g. for earliest, latest, cheapest), and of course predicates for interrogating the various database tables. The input TRL, on the other hand, consists of predicates representing the "natural" senses of the words used in the original utterance. The problem addressed by the database processing stage is that there is no simple one-to-one mapping from the latter to the former and non-trivial translation is necessary.

Consider the query "What afternoon flights are there from Boston to Denver on Delta?". This would be delivered to the database interface system as the TRL[1].

```
forall([A,B,C,D],
       impl(and(flight_AirplaneTrip(A),
            and(noun_noun_relation(A,B),
            and(afternoon_PartOfDay(B),
            and(from(A,airport#'BOS'),
            and(to(A,airport#'DEN'),
            and(on(A,airline#'DL')))))))),

            exists([E,F,G],
                and(future_event(E),
                    perform_action(E,display([A]),F,G))))
)
```

Paraphrasing, the input TRL reads "for every flight in the afternoon that goes from Boston to Denver on Delta ... let there be an event in the future in which we inform the user of it."

Note that declarative forms of request, such as "I want to fly from New York to San Francisco" are converted into imperative form ("Show me flights from New York to San Francisco") in earlier processing stages.

The output from AET is as follows:

```
forall([A,B,C,D,E,F,G,H,I,J,K,L],
       impl(and(db_flight(A,C,'BOS','DEN',B,D,E,'DL',F,G,
                          H,I,J,K,L),
            and('<'(B,1800),
                '>'(B,1159))),

            exists([O,P],
                and(db_time_to_printable_time(B,O),
                and(db_time_to_printable_time(D,P),
```

[1]These TRLs have been slightly edited to improve readability — note that type information like `airport#'BOS'` is in fact delivered in a somewhat different form; see section on object typing on page 215

```
                display_tuple([flight#(A),
                               ['Flight code','Day code',
                                'From','To','Departs',
                                'Arrives'],
                               [E,C,'BOS','DEN',O,P])])))
)
```

Which contains only structures meaningful to the database and the various executable predicates. Note, for example, translation of the concept "afternoon flight" into "flight departing after 11:59am and before 6:00pm".

Executing this sentence (which essentially defines a program for interrogating the database) results in the output:

```
 ============= ========== ====== ===== ========== ==========
| Flight code | Day code | From | To  | Departs  | Arrives  |
 ============= ========== ====== ===== ========== ==========
| DL841/DL845 | DAILY    | BOS  | DEN |  3:25 pm |  7:45 pm |
 ============= ========== ====== ===== ========== ==========
```

The next two sections, 15.2 and 15.3, provides a brief introduction to AET, linguistic domain theory construction and a summary of results obtained at the end of the project.

15.2 Translation and the Domain Theory

In this section we take a whistle-stop tour of AET and give an overview of the Linguistic Domain Theory used to define the translation scheme.

15.2.1 Abductive Equivalential Translation

This is an highly simplified outline of the AET mechanism, intended purely to give the reader some idea as to how this part of the system works. For more detail and fully worked examples the reader is referred to Rayner (1993).

The AET process works by incrementally translating an input TRL into an executable form according to a set of equivalences defined in a Linguistic Domain Theory (i.e. the set of equivalences, descriptions of functional relationships between various predicate arguments, and auxiliary implications).

What should be stressed is that the AET formalism makes for elegant, declarative and concise mapping between first order theories. The translation can occur in natural stages and handles more complex situations without great difficulty. Moreover, the approach seems to scale well: this project has significantly expanded the size of the LDT without having caused any noticable increase in processing time.

Translating Logical Formulae

The basic idea behind AET is that a sentence

$$P_1 \wedge Rest$$

can be translated into

$$(Q_1 \wedge Q_2 \wedge Q_3) \wedge Rest$$

provided we have an rule[2] to say that

$$(P_1 \wedge P_2 \wedge \ldots \wedge P_n) \Leftrightarrow (Q_1 \wedge Q_2 \wedge Q_3)$$

and we can prove that $P_2 \wedge \ldots \wedge P_n$ holds assuming $Rest$. The $Q_1 \wedge Q_2 \wedge Q_3$ may well go on to be translated further or take part in the translation of other terms in the resulting sentence. Translation stops once no more progress is possible. Note that there are translation schemata to handle all the standard logical connectives (conjunction as above, disjunction, implication, negation and quantification), but we need not go into detail here.

For efficiency reasons, a backwards chaining theorem prover is employed to decide the condition of applicability $P_2 \wedge \ldots \wedge P_n$ for the equivalence above. Implications (Horn clauses) derived from the equivalences in the LDT form the greater part of the inferential rules used to decide whether the condition is met or not.

A (simplified) example of an equivalence in the ATIS LDT specifying the (partial) translation of the concept "flights in the morning" is

$$\begin{array}{rcl} in(flight\#F, PartOfDay) & \wedge & morning(PartOfDay) \\ & \Leftrightarrow & \\ departure\_time(flight\#F, time\#DepTime) & \wedge & (DepTime < 1200) \end{array}$$

which reads as "saying F is a flight in $PartOfDay$ where $PartOfDay$ is a morning is equivalent to saying that $DepTime$ is the departure time of flight F and $DepTime$ is less than 1200 hours. In turn, we might have another equivalence

$$departure\_time(flight\#F, time\#DepTime) \quad \Leftrightarrow \quad co\_flight : [id = F, dept = DepTime]$$

to connect a flight and its departure time through the conceptual flight relation (for an explanation of the [...] slot notation and $name\#Object$ type scheme see below).

It is likely that multiple co_flight relations all keyed on the $flight\#F$ object will be generated in the course of translation (e.g. for $from$ and to constraints in the query). A simplifier is employed which can collapse them into a single instance since it knows that there is a functional relationship from the id slot to all the others for conceptual flight terms. The simplifier also uses implications derived from the equivalences in the LDT to discard terms which can be deduced from their context.

Unsuccessful translation attempts are detected by the presence of non-executable predicates in the TRL. Otherwise the term is reordered to manimise the amount of database searching (again using knowledge of the functional relationships between arguments in various predicates) before being executed.

Implicit Quantification

Strictly speaking, AET requires that all but variables with global, universal scope need to be explicitly quantified. On the other hand, it is often obvious from an equivalence what the appropriate quantification for variables is.

[2]Note that we use rules in a left-to-right fashion only in order to keep things tractable.

The general rule is that equivalences of the form

$$p(X,Y) \quad \Leftrightarrow \quad q(Y,Z)$$

are interpreted as meaning

$$\exists X.p(X,Y) \quad \Leftrightarrow \quad \exists Z.q(Y,Z)$$

with free variable Y being treated as universally quantified as per usual.

Due to the nature of the way AET works, non-atomic existential equivalences must split into separate universal and the atomic existential cases, so

$$\exists X,Y.p(X,Y,Z) \wedge q(X,Y,Z) \quad \Leftrightarrow \quad r(Z)$$

becomes

$$\begin{aligned} p(X,Y,Z) \wedge q(X,Y,Z) \quad &\Leftrightarrow \quad new(X,Y,Z) \\ \exists X,Y.new(X,Y,Z) \quad &\Leftrightarrow \quad r(Z) \end{aligned}$$

The LDT constructor, however, need not be concerned about these issues since preprocessing automatically converts equivalences into canonical form. In fact, the only time explicit quantification is necessary is when the right hand side of a rule has a local universally quantified variable (a situation which has not, so far, arisen in the ATIS LDT).

Object Typing

To increase efficiency a simple (flat) syntactic type-scheme is now in use. At an early stage in translation conceptual objects denoting such things as flights are tagged with their type. So a clause such as `flight(F)` in the input TRL will be translated into `F=flight#Id`.

It should be kept in mind that the type scheme is (a) purely notational and (b) just a means of making explicit object types that are implicit at the linguistic and database levels. In other words, this form of typing information is only acquired and employed in the LDT and has no bearing in other phases where it would, more than likely, get in the way.

Efficiency is improved since Prolog's unification will prevent consideration of equivalences whose predicates contain slots with incompatible types (e.g. for separating cases such as 'flights from..." and "fares from...'). Use of explicit typing also helps improve readability of equivalences:

```
equiv(economy_fare, regularize,

        economy(fare#(Id, OneWayOrReturn)),

        and(co_fare:[
                fare_id=fare#(Id, OneWayOrReturn),
                fare_basis_code=
```

```
                    farebasis#DBFareCode ],
            db_fare_basis:[
                fare_basis_code=DBFareCode,
                economy='YES'' ])
).

equiv(economy_flight, regularize,

        economy(flight#Id),

        and(co_flight:[ flight_id = flight#Id ],
        and(co_flight_fare(flight#Id,
                           fare#(FareId, OWorR)),
            economy(fare#(FareId, OWorR))))
).
```

15.2.2 Summary of the ATIS Linguistic Domain Theory

The ATIS LDT comprises some 459 equivalences divided into seven groups[3]. Translation is restricted to one group at a time, the next group only being applied when no further progress is possible with rules from the current group. The idea is that each rule group canonicalizes one aspect of the sentence under translation: groups exists for identifying type information (69 equivalences), pre-processing relational structures (32 equivalences), regularizing linguistic forms (this is the largest group with 258 equivalences), handling temporal relationships (21 equivalences), display of information (24 equivalences), final translation from the general conceptual level to the database level (10 equivalences), and optimisation (23 equivalences); the remainder consist of rules for format conversion etc.

Rules in each group are largely stereotypical and by and are generally fairly straightforward, although the formalism is well equipped to deal with cases where more sophistication is required (e.g. for informative error feedback) and there are subtle cases where the LDT coder must exercise care. Here are examples of typical equivalences from each class:

The LDT is constructed such that translation occurs in three roughly distinct stages. First of all the untranslated TRL is converted from using predicates denoting natural word senses to an intermediary conceptual level (this is also the point at which object types are identified). Then the conceptual form is translated into a simpler version in which everything has a direct database analogue. Finally, the canonicalised conceptual level sentence is converted to the database level in which all predicates denote either database lookups or executable procedures (e.g. for comparison or formatting of dates).

- **Typing**

```
equiv(flight_arrives, types,
```

[3]Of these 110 are domain independent "legacy equivalences" from previous projects. It is believed that this number could be reduced given our now much improved understanding of the domain.

```
        arrive_TurnUp(Event, flight#Id),

        and(Event = arrival_event#(flight#Id),
            co_flight:[ flight_id=flight#Id ])
).
```

This equivalence (given the name "flight_arrives" in the "types' rule group) translates the verb "arrive" in the context of a flight by connecting `Event` to the arrival of that flight; the `co_flight:[ ... ]` part simply ensures that the translation includes reference to some *particular* flight. The `name#Object` forms simply declare `Object` to be an instance of the type `name`.

- **Relational Pre-processing**

```
equiv(on_relation, regularize,

        on(X, Y),

        r(X, on, Y)
).
```

Here we are just transforming the structure of a two place relation `on` into the canonical three place relation `r` which contains the name of the original. This allows us to generalise over relations which are interchangable in certain contexts (see the next example).

- **Regularisation**

```
equiv(flight_on_with_relative_run_by_for_airline,

        regularize,

        r(flight#Id, R, airline#Airline),

        co_flight:[ flight_id=flight#Id,
                    airline_code=airline#Airline ]

) :- member(R, [on, with, relative, run_by, for]).
```

Translates any phrase of the form flight on/with/for/relative to/run by some airline by connecting the flight and airline objects `Id` and `Airline` via the appropriate slots in the conceptual flight relation. Without the generalised `r` relation we would need a separate equivalence for each form.

- **Temporal Relationships**

```
equiv(date_before_seconds_granularity, temporal,

        date_before3(Type, [ Date1, seconds ],
                           [ Date2, seconds ]),

        date_before4(Type, seconds, Date1, Date2)
).
```

This is a straightforward translation of an intermediate term into a canonical form. The handling of time is overly general in the current LDT: the model of time used in the database is very simple, consisting of just days of the week and integers to represent time on the clock; on the other hand the inherited temporal model comes from a domain with a much more sophisticated notion of time.

- **Displaying Information**

```
equiv(display_airport, regularize,

        display_format(airport#Airport_Code, Columns,
                       Data),

        and(db_airport:[ Airport_Code, Airport_Name,
                         Airport_Location ],
        and(Columns = [ 'Code', 'Full name', 'City'' ],
            Data    = [ Airport_Code, Airport_Name,
                        Airport_Location ]))
).
```

This equivalence says how to display an airport: columns are headed “Code”, “Full name” and “City”. The data for the last two colums is obtained by performing a database lookup.

- **Translation to Database Predicates**

```
equiv(co_ground_service_to_db, db_connect,

        co_ground_service(Co_Ground_Service, Co_Cost),

        and(Co_Ground_Service =
                transport#(Co_City, Co_Airport, Co_Type),
        and(Co_City           =  city#DB_City_Code,
        and(Co_Airport        =  airport#DB_Airport_Code,
        and(Co_Type           =  type#DB_Transport_Type,
        and(Co_Cost           =  cash#DB_Ground_Fare,
            db_ground_service(
                DB_City_Code,
                DB_Airport_Code,
```

```
                    DB_Transport_Type,
                    DB_Ground_Fare ))))))
).
```

Here we translate the conceptual airport ground service (e.g. bus, taxi) representation into the appropriate database lookup. The conceptual level objects are all explicitly typed, whereas the database records are not. Note also that the ground service key `transport#(Co_City, Co_Airport, Co_Type)` is structured so as to uniquely define a ground service instance. The practice of using unique keys whenever possible has significant efficiency benefits since we can merge predicates with identical keys into a single term.

The stereotypical nature of the majority of equivalences suggests that an easy to use form-filling tool or some-such could be used by inexpert programmers to develop large parts of an LDT for a new domain. Indeed, it has become apparent that many equivalences connecting different database relations could be automatically generated from a suitable description of the database schemata. We therefore believe that we are in a position to be able to build tools which would greatly aid in porting to new domains.

The remainder of this section consists of an highly abridged introduction to AET, a summary of achievements including coverage, reorganisation of the LDT, preprocessing stages for a simple type and slot filling mechanism, handling of error predicates, use of preferences in selecting applicable equivalences, sticky defaults for context dependent dialogue processing, and improvements to the robustness and usability of the testing software.

15.3 What Has Been Achieved

15.3.1 Scores on the New Context Independent Repcorpus (Small and Full-Size versions of Database)

Development of the system has been based around increasing the coverage of queries taken from a representative corpus, that is, a subset of the full corpus selected to contain instances of the great majority of utterance types. Initial development was conducted using an 175 example repcorpus covering 1101 utterances. For this part of the project a larger 766 example repcorpus was constructed, this time covering 5022 utterances. Since the utterances in the repcorpora are selected by frequency, the smaller repcorpus is effectively just the front part of the new version.

It should be pointed out that these repcorpora only consist of context independent queries; that is, those whose meaning does not depend upon the surrounding dialogue. So, while an query such as “What are the flights from Boston to Atlanta on Saturday?” is regarded as context independent, “Just the first class fares.” is not since it clearly refers to a preceeding reply. Context dependent processing is discussed elsewhere in this document.

At the time of writing (2nd January 1997) a success rate of 82.8% has been achieved on the new repcorpus. This shows an improvement in excess of 25% compared to

when the project started in October where the score was somewhere in the mid fifties. It is true that on the older representative corpus scores in excess of 90% were being recorded, however the new corpus is nearly five times as large and presents a much broader range of problems (the larger corpus contains a much higher proportion of low-frequency query types). Still, the success rate with the new corpus after the first 1100 represented queries is 100%, while after 175 examples is 94.5% which strongly suggests that general coverage has been greatly improved.

The current implementation holds the database as a collection of Prolog records. For pragmatic reasons we have been working with a restricted subset of the database consisting of the thirteen most important relations ranging over ten cities. The full size database consists of a further fourteen relations ranging over about fifty cities. Tests were made on a full size version of the database and, apart from the relatively long time needed to load the image, no significant difference in either speed or accuracy of responses was observed. It is intended that future versions of the system should make use of a proper database engine which should drastically reduce the size of the Prolog image and improve performance.

Coverage currently includes:

- flights;
- times of arrivals and departures (including earliest/latest);
- periods during the day such as morning and night and dates including days of the week and/or particular month days;
- stops for flights, meals served on flights, fares and fare classes (e.g. first, business);
- prices (including cheapest, most expensive);
- ground transport available at airports;
- airlines; and
- and informative error recovery.

This accounts for the majority of the most frequent types of request in the corpus, although there are obviously some gaps. Perhaps the most significant of these is that the system currently has very little provision for handling queries asking for several different types of information. Thus “Show me the fares for nonstop flights from Boston to Denver” causes no problem (only the fares are being requested), but something like “Flights and fares from San Francisco to Dallas” will cause the system to balk since it is a request for two distinct types of data, namely flight information and fare information. Although it is no great problem to extend the database system to handle this kind of query, it turns out that some support upstream is also necessary (e.g. for “flights *and* fares...” or even “flights *with* fares...”). This is the responsibility of the resolution phase which is discussed in another section.

15.3.2 Reorganisation and Recoding of Equivalences

Efforts have been made to regularize and simplify the structure of the equivalences comprising the linguistic domain theory (LDT) defining the mapping from untranslated TRLs to translated TRLs which can be "executed" to produce a response from the database.

The general scheme used is to translate first into a canonical 'conceptual" or "open" form, then into an executable "closed' form, and finally perform various simplifications. Conceptual level predicates are intended to be "open" versions of their closed database counterparts (the difference being that, for example, while a conceptual predicate might refer to flights from Boston to Denver, the corresponding database predicate can only refer to such flights as are recorded within the database, and to those by the various codes used in the database model). Conceptual predicates now all have names starting with "`co_`'; database predicates start with "`db_`'. Otherwise corresponding predicates have suffixes with names identical to those used in the defining database schema.

Slot/Value Notation

Since many predicates take a large number of arguments of which only a few are generally of relevance in any particular case, a slot/value notation has been implemented. Thus, it is now possible to rewrite, say

```
co_flight(Id, _, From, To, _, _, _, _, _, _, _, _, _, _, _, _)
```

as

```
co_flight:[flight_id=Id, from_airport=From, to_airport=To]
```

where the latter is expanded to the former by the pre-processor.

In the former case, anonymous variables would generally be eschewed in the LDT and instead proper names would be used to aid documentation of the code:

```
co_flight(
    Id, Day, From, To, Departure_Time, Arrival_Time,
    Airline_Flight, Airline_Code, Flight_Number,
    Aircraft_Code_Sequence, Meal_Id, Stops,
    Connections, Dual_Carrier, Time_Elapsed
)
```

This is clearly too cumbersome and long winded. On the other hand, while the amount of typing required is not greatly reduced if anonymous variables are used in the LDT, the need to remember place values is relieved and the LDT becomes much easier to read.

Inclusion of Error Predicates in the Translated Form

One of the more important aspects of the user interface is provision of helpful error messages when something goes wrong. Whilst it is obvious when the system cannot

translate an utterance, there are other classes of error which hitherto have not been handled in a terribly satisfactory way (the most common response being to fail relatively quietly).

To improve matters we have taken advantage of the fact that many kinds of data errors, such as reference to flights that do not exists or at least are not recorded in the database, can be identified at translation time. In such cases we include a special error predicate in the translation giving details of the nature of the problem. Before a translated query is executed, it is checked to see whether any such error terms form part of the translation. If so, execution is abandoned and an helpful message printed on the basis of what is contained in the error term. This generally occurs in situations like the one mentioned above. It cannot cover all situations, however: if the user mentions, say, a city that the system has no knowledge of then it cannot deduce that what was mentioned was indeed a city. On the other hand, if the user mentions London and the system knows that London is a city, but one which is outside the sphere of the database, then the error term mechanism can be usefully applied. [Another benefit of the error mechanism is that sticky defaults (see later) acquired from the problem term can be withdrawn if necessary].

Use of preferences to simplify application of default cases

Applicability of equivalences during the translation process is determined by deciding whether appropriate preconditions have been met by the formula being translated. This processing is carried out by the theorem prover. Unfortunately, proof of inverted goals is generally quite difficult in domains as broad as ATIS and indeed the current implementation of the theorem prover often fails to prove such goals when it should. This is a problem since there are relatively common situations where it is useful to be able to assume that certain things are false if they are not present (consider "I would like an early morning flight" vs. just "I would like a morning flight"). In order to get around the problem, we have exploited the preference mechanism used in several parts of the system to guide AET into testing more specific equivalences first (so in our example, the system would first try to translate "morning" with the "early morning" equivalence before going on to the more general one). Once an equivalence has been successfully applied, the system commits to that choice. It is felt that this approach is good in that it is robust, easy to understand, and avoids the efficiency problems of having to deal with negation more than is strictly necessary.

Improved Interface to Existing Testing Software and Improved Robustness in Testing Facilities

Due to rapid changes in various parts of the system, there has been a drift between the interface to the main parts of the system, the structures output, and the structures and interfaces expected by the testing software. As is often the case with such things, problems have been solved by successive layers of patches, leading to a somewhat baroque suite of testing facilities. In order to ameliorate the problem, we have developed a test-debug-loop interface that hides much of the underlying complexity behind a more user-friendly menu driven interface. We have also put some effort into making testing

procedures robust (since the system is generally in a state of flux, sometimes an unnoticed bug introduced by a change can cause a test run to crash at some point. The robustness facility now spots when this happens and quietly restarts the test at a point after where the crash occurred). As with much of the above, the next iteration of the project will include development of a reworked set of testing facilities taking advantage of experience gained from this and previous projects.

15.4 Context-Dependent Database Query

15.4.1 Overview

ATIS dialogues, though limited in their use of vocabulary, exhibit a very wide range of context dependent phenomena. Although there is surprisingly little use of pronouns, there is a huge variety of elliptical constructions, plus various kinds of contextually restricted noun phrases. Processing of dialogues in the SLT system proceeds through several stages. For example, consider the following dialogue from the Atis2 corpus:

> I would like to travel from Boston to Denver
> I would like the cheapest flight
> One way

The effect of QLF-to-QLF translation (as described above) is to convert the first two sentences to equivalent direct requests. The dialogue thus becomes equivalent to:

> Show all flights that travel from Boston to Denver
> Show the cheapest flight
> One way

We now have two further tasks. To interpret the second utterance correctly, we do not want the cheapest flight on any route, but the cheapest flight from Boston to Denver. To interpret the third utterance, we need to form a query along the lines of *Show the cheapest one way flight*, and then restrict *flight* to *flight from Boston to Denver*. Both tasks require the use of contextual information, not just domain knowledge.

In the database system the two tasks are handled at different stages. Contextual restriction (e.g. restriction of *flight* to *flight from Boston to Denver*) is handled by a *sticky default* mechanism. Formation of queries from ellipses, and traditional reference resolution problems (such as interpretation of pronouns) are handled by a separate reference resolution component.

The two mechanisms both have advantages for their particular tasks. The sticky default mechanism is essentially a slot filling mechanism. By choosing database arguments as slots, this ensures that information is in as canonical a format as possible. For example, *from Boston* and *leaving Boston* are both mapped to the value *Boston* in the departure slot. A canonical format makes it possible to recognise when a new value should replace an old without employing complex reasoning (for example, a departure slot cannot be filled by both Boston and Philadelphia simultaneously).

In contrast, reference resolution is done at the level of QLFs. Although this means that information is in a less canonical format (though more canonical than e.g. the

surface string), this is necessary for the kind of phenomena reference resolution deals with. Consider the following contrasting pairs of dialogues:

Show me flights from Boston on American.
I would like to go to Denver.
What is the ground transportation?

Show me flights to Denver on American.
I would like to fly from Boston.
What is the ground transportation?

Show me the cheapest fare for a flight from Boston to Denver
How about Altanta?

Show me the flight from Boston to Denver with the cheapest fare
How about Altanta?

In the first pair of dialogues, different cities are more or less recent, and hence salient. This means that the preferred reading for the first case is *ground transportation in Denver* and in the second, *ground transportation from Boston*. In the second pair of dialogues, different nouns head the questions: *fare* in one case, *flight* in the other.

The kind of recency information or structural information which these examples require would not normally be found in contexts formed purely by slot filling, although extra slots can be added to deal with specific examples (for example, the BBN system (Miller et al. 1996) can deal with the second pair of examples since they add a special slot to contain the last head noun mentioned).

15.4.2 Reference Resolution Component

The reference resolution component was built over a long period, and is described in the CLE book (1992). The treatment of ellipsis is described in Crouch (1995). Resolution is rule based with hand-coded preferences. The resolution component is comprehensive with a treatment of common kinds of definites, pronouns, demonstratives and ellipses.

The major change to reference resolution in the new system is the addition of rules to deal with highly elliptical utterances consisting of one or more prepositional phrases, a single noun phrase, or an *ing* phrase such as *flying first class*. Consider, for example, the treatment of the following example:

Show me flights from Boston to San Francisco on Continental
On Saturday morning
To New York

The last two utterances need to be expanded into suitable queries. The first is treated by extracting a salient noun phrase *flights from Boston to San Francisco on Continental* and replacing it by the head noun, *flights* with the restriction *on Saturday morning*. The result after reference resolution is thus:

Show me flights on Saturday morning

This is then restricted by sticky default processing to a query equivalent to

> Show me flights on Saturday morning from Boston to San Francisco on Continental

Similarly, *To New York* results in replacement of the noun phrase *flights on Saturday morning* with *flights to New York*. This is then restricted by the sticky defaults to:

> Show me flights to New York on Saturday morning from Boston on Continental

The final query includes all the prior constraints except for *to San Francisco*. This has been achieved by reference resolution providing a minimal, underspecified, query which just includes the old head noun plus the new constraint, and sticky defaults providing the appropriate constraints from the prior context.

So far, the idea of using reference resolution to produce minimal queries has only been used in the new rules extending the coverage to highly elliptical utterances. In future work, it is likely that this idea will become much more widely used throughout the reference resolution component, giving much simplified rules, and a clearer demarcation between the work of reference resolution and sticky default processing.

15.4.3 Sticky defaults

It turns out that a good deal of the context of an utterance in a dialogue can be obtained from the translated forms of the preceeding utterances. This context can be extracted and used to fill in "holes" in succeeding translations. For example, if the user asks for "flights from Boston to New York", the system will pick out (from = "Boston") and (to = "New York") as part of the context for later translations. Then, when the user follows up her question with "flights after 5pm", "Boston" and "New York" will be used to fill the to and from slots of the flight relation in the final translation. Furthermore, the new constraint on the flight times being before 5pm will also be extracted and applied in later translations if appropriate.

In the case of dialogues where the user changes some part of a request, such as in "New York to San Francisco on June twenty first", "How about on the twenty second?", we find a clash between the context (day = "twenty first") and the information in the latest request (day = "twenty second"). This problem is resolved by simply replacing the old context for the day slot, in this case, with the new value specified in the request whenever there is a conflict. Currently this is done in a rather simple-minded way which does not take into account refinements of constraints on particular slot types. For instance, "Flights after noon", "Before 6pm" will end up with only the default (departure_time < 6pm), rather than what was probably intended by the speaker (noon < departure_time < 6pm). We intend to greatly improve upon this in future work (see later section). However, for the present, the worst that can happen is that the user receives slightly more information than expected.

This approach is surprisingly effective and helps simplify the task of the more complex reference resolution stage which precedes AET. More work remains to be done, of course. The most notable problems are (1) that transitive dependencies such as "flights

to Oakland after 7pm", "show me the fares", where the second query is referring to fares for flights *after 7pm*, are currently not dealt with and (2) that the currently fairly simple minded way of extracting context relevant to a slot can cause the size of the sticky defaults to explode, each containing much that is irrelevant or unhelpful.

15.4.4 Development suites

The work on reference resolution has been empirically driven from ATIS 2/3 dialogues. To ensure the most important phenomena were treated first we developed two corpora of representative dialogues. The most important contains around fifty representative dialogues for different anaphoric phenomena, ordered according to how large a set of examples the dialogue represents. This set was created by annotating a set of approximately 1400 ATIS2/3 dialogues according to the following features:

1. class of anaphor e.g. bare noun, pronoun, demonstrative
2. way context is used e.g. full, partial
3. requirement for real world reasoning
4. kind of antecedent e.g. linguistic, bridging, database entity, none.

Equivalence classes were created for dialogues containing the same annotation for one or more of their anaphors, and a representative dialogue was picked from each equivalence class. Since a dialogue can contain more than one anaphor, and hence can appear in more than one class, the selection program ensures any one dialogue appears no more than once in the representative corpus.

To provide some check that the procedure above is providing good candidates for representative dialogues, we also used a second development corpus based on gathering together around a hundred of the most frequent dialogue kinds. This was based merely on syntactic similarity between non-initial utterances in dialogues. As might be expected, short utterances such as *to* ⟨*cityname*⟩ appear highly ranked.

Chapter 16

Common Language-Speech Issues

David Carter, Jaan Kaja, Leonardo Neumeyer, Manny Rayner, Fuliang Weng and Mats Wirén

In this chapter we cover a number of issues arising from the need to interface the Decipher recognizer with the CLE. These issues include the number of sentence hypotheses that should be processed, covered in Section 16.2; the coverage of compound nouns in languages like Swedish, where they are productive (Section 16.3); and the use of acoustic scores in parsing and disambiguation, already covered in Sections 6.2.3 and 6.4.3. We begin, however, with a description of the interface itself.

16.1 The Speech-Language Interface

The CLE converts N-best sentence hypothesis lists provided by Decipher into word lattices; see Figure 6.1 on page 68 for an example. This is done by initializing the lattice to be the sequence of words in the first hypothesis, and then, for each subsequent sentence hypothesis in turn, adding as few edges as possible to it so as to provide a path corresponding to that hypothesis. As each edge is added, it is assigned the acoustic cost (represented as the shortfall from the score of the top hypothesis) of the current sentence hypothesis. This lattice forms the basis of the analysis chart (see Section 5.2).

Usually, each new sentence hypothesis after the first can be catered for by adding only one or two new word edges to the lattice. Thus most of the redundancy inherent in the N-best list is factored out, allowing parsing to be much more efficient than if each sentence hypothesis were processed independently. Often, the word lattice will license a few paths not included in the N-best list; for example, an N-best list consisting of the three hypotheses “one way”, “one day” and “some way” would give rise to a lattice in which either “one” or “some” could be followed by either “way” or “day”, thus licensing the sequence “some day”. This extra generality, when it makes any difference at all, is usually an advantage, as sometimes one of the additional paths is the correct

N	%Correct	Shortfall	%Correct	%Gain
1	60.6	0	60.8	+0.2
2	70.3	102	70.0	-0.3
3	73.4	165	73.1	-0.3
5	76.6	259	78.2	+1.6
10	80.5	427	81.6	+1.1

Table 16.1: "Correct" N-best lists for various N and corresponding shortfalls

one, and an analysis is found and selected for it.

A number of subtleties arise in constructing the lattice. Firstly, some sentence hypotheses can be ignored: if the words in a hypothesis have the same sequence of semantic classes as a better-scoring hypothesis (e.g. "Show me flights to Oakland" when "Show me flights to Atlanta" scores better) then it cannot give rise to the winning analysis, so is pruned out at this early stage. Secondly, filled pauses ("um", "er") are simply deleted before the word sequence is added to the lattice. Thirdly, for some languages, especially French, it is convenient for the recognizer to treat as words some units which syntactically are multiple words, and vice versa: e.g. in French, "d'American Airlines" is recognized as the two units "d'American" and "Airlines", whereas the CLE treats them as "d' " (a contraction of "de") followed by the lexical item "American Airlines". The necessary separating and joining of lexical items is done by applying the CLE's spelling rule mechanism (see Appendix A) just as for text input.

16.2 The N-best interface: what should N be?

Currently, the SLT system uses an N-best interface with N fixed at 5. We have experimented with different values of N, reported below, and also with the idea of processing not a fixed number of sentence hypotheses, but all sentence hypotheses whose score is within a fixed threshold of the best one.

A priori, one might expect that a hypothesis in, say, seventh position in the N-best list will be more likely to be correct the closer its score is to the top hypothesis in the list. We tested for this possibility using N-best lists produced by Decipher for a corpus of 1000 ATIS-3 test N-best lists not previously used for system development.

For each of the values of N listed in the first column of table 16.1, we show in column two the percentage of utterances for which the reference version was in the top N, making the N-best list a "correct" one in the sense that it contains the correct word sequence. The "Shortfall" is the shortfall threshold (i.e. the maximum allowable shortfall from the recognizer score for the top hypothesis) which causes the same number of hypotheses to be considered over the corpus as a whole as that value of N, and would therefore be expected to require similar overall processing times. Irrespective of the shortfall, no more than 20 hypotheses were considered, as to do so could lead to rather slow processing. Column four shows the percentage of utterances for which the reference version fell within this shortfall of the top one. The "gain" is the difference in correctness percentages between the shortfall and fixed-N methods.

N_1	N_2	N_1 time	N_2 time	N_1 better	N_2 better	Ties	Gain
1	2	0.76	0.87	17	45	7	+2.8%
2	3	0.87	0.96	7	17	5	+1.0%
3	5	0.96	1.14	15	19	4	+0.4%
5	10	1.14	1.58	28	26	3	-0.2%
10	20	1.58	2.48	26	17	4	-0.6%

Table 16.2: Analysis performance for different N values

Perhaps surprisingly, no clear advantage exists for the shortfall method. The apparent advantage for N=5 is within the bounds of statistical variation, and in fact in this particular sample of data, the fifth position contains, by chance, relatively few correct hypotheses: ten, compared to 22 in fourth position and 15 in sixth.

This confirms our decision to use the fixed-N method, partly for reasons of simplicity (e.g. it is easier to display and explain a fixed-length list) and partly because fixing N is likely to lead to less variation between the language analysis times for different utterances. Furthermore, there is no guarantee that the mere *presence* of the correct utterance in an extended N-best list will lead to it being *selected* for translation; thus even if the 1.6% advantage for N=5 in our sample reflects a real underlying difference, it may not lead to much improvement in overall system performance, since the crucial extra 1.6% of correct hypotheses are by definition lower than position 5 in the list and are therefore competing with a lot of other acoustically superior possibilities.

In fact, the same observation applies to the idea of increasing N in a fixed-N interface; doing so will obviously lead to more correct hypotheses being considered (see the increasing figures in column two of Table 16.1) but there is no guarantee they will be selected for translation. There is, presumably, a value of N at which selection accuracy stops increasing or even goes down.

To assess this, we looked at the relative performance of the English system, for the different values of N given above plus N=20, when run on the same corpus of 1000 ATIS-3 test N-best lists as used for the first experiment. For each neighbouring pair of N values, when the strings for the selected QLFs differed, the strings were compared, and the one which seemed closer in meaning to the reference sentence was marked. When a QLF was produced for only one value of N, a decision was made on whether the string for that QLF suggested it would be likely to produce a translation that would be better than nothing: that is, that it would lead a hearer to provide some useful information, rather than misleading information or no information. Sometimes, there was a tie, when both outcomes seemed equally good or equally bad.

In table 16.2 we show the results of comparing neighbouring values of N, and in particular the "gain" for each pair of values: the improvement in overall source side (speech recognition and language analysis) performance to be expected from moving from $N = N_1$ to $N = N_2$, measured as the number of times N_2 gives a better result minus the number of times N_1 is better. The times shown are the average number of CPU seconds needed for language analysis on a Sparc Ultra-II, using the conventional (non-"anytime") LR parser. Because of statistical variation, the gain figures are

only reliable to within about 0.6%. Also, it would have been slightly more informative (though more time-consuming) to compare translations rather than selected source strings. Nevertheless, the trend is clear: increasing N to 3 gives a definite advantage over $N = 1$ and $N = 2$, and $N = 5$ may well be better still, but thereafter performance does not improve and may even worsen, while processing time (unsurprisingly) continues to increases.

16.3 Handling Compound Nouns in Swedish

This section describes and evaluates a simple and general solution to the handling of compound nouns in Swedish and other languages in which compounds can be formed by concatenation of single words. The basic idea is to split compounds into their components and treat these components as recognition units equivalent to other words in the language model. By using a principled grammar-based language-processing architecture, it is then possible to accommodate input in split-compound format.

16.3.1 Introduction

In many languages, including German, Dutch, Swedish, Finnish and Greek, compound nouns can be formed by concatenation of single nominals. For example, in Swedish "folk" and "musik" can be put together to form "folkmusik" (*folk music*), and this in turn can be combined with "grupp" to form "folkmusikgrupp" (*folk music group*), and so on. In most previously reported work, such as the widely publicized SQALE project, compounds have been treated in the same way as any other words. However, as the vocabulary size grows, the productive nature of compounding makes this kind of approach increasingly less feasible; the most obvious indication of the problem's seriousness is the magnitude of the out-of-vocabulary (OOV) rate. For example, in an experiment on SQALE training texts (Lamel et al. 1995:186), using 20 000-word lexicons for both German and English resulted in a 7.5 % OOV rate for German and 2.5 % for English. The German lexicon had to be extended to 64 000 words to obtain OOV rates similar to those of the 20 000-word lexicon for English.

Results like those quoted above strongly suggest that treating compounds in the same way as other words is not satisfactory. Two recent papers address this issue. Spies (1995) reports results on an isolated-word large-vocabulary German dictation application, in which the components of compounds, rather than the compounds themselves, were treated as units. Geutner (1995) describes a more elaborate method, also implemented for German, in which full morphological decomposition of words was used. Both authors report unspectacular but encouraging initial results.

This section describes and evaluates a simple and general solution to the handling of Swedish compound nouns, carried out in the spirit of the work reported in the two above-mentioned papers. Syntactically, semantically and phonologically, Swedish compound nouns are similar to English compound nominals except for the obvious difference: English orthography inserts spaces between components, while Swedish omits them. These observations suggested to us that Spies' strategy should be an appropriate way to attack the problem: splitting compounds into their components, and

treating these components as recognition units equivalent to other words in the language model. In contrast to Spies, however, our recognizer is for continuous speech, and is embedded in a spoken language understanding system. It is thus necessary not only to recognize, but also to reassemble and make sense of the split compounds, the means to do this being provided by the language-processing modules of the system.

We believe that our methods should handle verbal and other kinds of compounds equally well; however, since noun compounding in Swedish is more productive, and the other kinds of compound less common in our material, we have chosen here to concentrate on nouns. Inflectional morphology is a less serious problem in Swedish than in German (in particular, Swedish verbs are not inflected by either number or person). For this reason, we decided that full morphological decomposition *à la* Geutner would probably not justify the additional complexity introduced.

Our approach has been fully implemented within the Swedish-to-English version of the SLT system.

The rest of this section is organized as follows. Section 16.3.2 describes the corpus material used for othe experiments. Section 16.3.3 describes experiments involving the Swedish speech recognizer alone, and Section 16.3.4 describes further experiments on the full speech translation system. Section 16.3.5 presents our conclusions.

16.3.2 Corpus

The current version of the SLT system operates in the ATIS domain. For English, there is a carefully collected corpus of about 20 000 utterances. No corresponding corpus existed for Swedish when the current project started in 1995. We have gone through several iterations of creating successively more realistic Swedish versions of the ATIS corpus. The experiments described here were performed using Version 1 of Swedish ATIS (hereafter "ATIS-S-1"). A second version of Swedish ATIS, constructed since then, is described briefly in Section 16.3.5, and a third version is currently being collected.

ATIS-S-1 was produced by the following process. First, a set of about 5 000 original English ATIS utterances was randomly selected from the full English ATIS corpus. Four randomly selected subsets, each of about 2 800 sentences, were then each translated into Swedish by each of four different Telia employees. Finally, the resulting Swedish sentences were divided, roughly equally, between 100 native speakers of the Stockholm dialect of Swedish for reading and recording. In total, 11 275 sentences were recorded, of which 10 831 sentences were used for training and 444 held out for testing. The primary goal of this initial corpus collection effort was rapid creation of training material for a first version of the Swedish recognizer. A secondary goal was to provide basic text resources for use in the development of the Swedish language-processing modules.

The orthographic transcriptions of the ATIS-S-1 sentences were further processed to create two different versions of the text corpus. In the first, "split" version, all compound words, including numbers, were split into their components. In the second, "unsplit" version, only numbers were split. In both cases, the numbers were split because it would be futile to try to list them in the lexicon; the same approach was taken in the German SQALE experiments (Lamel et al. 1995:186).

	Split	Unsplit
WER with respect to compound components (method 1)	**7.9** %	8.2 %
WER with respect to full compounds (method 2)	8.3 %	**8.7** %

Table 16.3: Word-error rates obtained in the experiments.

The split and unsplit versions of the ATIS-S-1 text were used to train two different versions of the Swedish recognizer. The two versions of the recognizer differed only in terms of vocabulary and language model. The recognition vocabulary consisted of the set of all surface words in the relevant version of the corpus. The bigram language model was calculated directly from the corpus, without, for example, backing off surface words to classes.

16.3.3 Speech-Recognition Experiments

To compare the split and unsplit approaches at the recognition level, we performed two experiments with respect to word-error rate (WER), using data from the full set of 444 test sentences with 3 584 unsplit words and 3 758 split words. In one experiment, split training data and a split lexicon were used for language modeling; in the other, unsplit training data and an unsplit lexicon were used. The results are shown in Table 16.3.

Since the total number of words is different in the split and unsplit cases, the WER with respect to compounds can be measured in two ways. More specifically, the following two methods for calculating the WER were used: In the first one, corresponding to the first row of Table 16.3, a splitting function, which (for the purpose of the experiments) is used for mapping compounds to their components, was applied to both the hypotheses from the recognizer and to the references. (This function modifies only the unsplit data.) We then compared the newly formed hypotheses and references to get the WER. Thus, in this case the WER was calculated with respect to the compound components.

In the second method, corresponding to the second row of the table, the same splitting function, but with mappings of numbers removed, was used in the reverse direction to map all the compound components in both hypotheses and references back to their compounds. The result was then used for computing the WER. Thus, in this case the WER was calculated with respect to the full compounds.

From the point of view of language processing, it is the main (bold-faced) diagonal that is of primary relevance, since what we want to compare are recognizers that output either split or unsplit words. These figures show a modest improvement in WER. The other diagonal has been included to provide a fair comparison from the point of view of speech-recognition performance.

As for the unsplit case in method 1, we have the methodological problem that com-

pound words as well as their components are in the recognizer lexicon. There is thus a good chance that the recognizer will output the components, whereas the reference contains the compound. This will then be counted as one substitution followed by one or more insertions. It can be argued that the confusion between a compound and its components is not a major error. Method 1, applied to the unsplit case, removes this ambiguity and gives a performance figure that can be compared with the split case.

Method 2 can be said to simulate a language-processing system in the sense of reassembling the compounds. In most cases, this reverse mapping is straightforward, but there are cases in which a potential compound does not actually constitute a compound where it occurs in a sentence. For example, the temporal noun phrase "måndag eftermiddag" (*Monday afternoon*) has a corresponding compound "måndageftermiddag". However, the two forms differ in meaning (and are also prosodically distinct). Since method 2 creates a compound from every sequence of words that in some context could be a compound, it does not take this difference into account, and in this sense the figures in the second row of Table 16.3 are imperfect.

As a comparison, the WER in the unsplit case *without* the reverse mapping is 9.3 %. This number is relevant to tasks like dictation, where a confusion between a compound and its components would be considered an error.

16.3.4 Split vs. Unsplit Compounds in Speech Understanding

The results in Section 3 show that compound splitting produced a modest improvement in the recognizer's WER. In the context of a speech-*understanding* system like SLT, however, the most relevant criterion for success is the effect on end-to-end performance. Another question of practical importance is the extent to which the language-processing modules need to be altered to accommodate input in split-compound form.

End-to-end performance evaluation

To test the effect of compound splitting on end-to-end system performance, we used the 444-sentence test set as input to two experiments involving language processing as well as speech recognition. The two sets of N-best speech hypothesis lists were each processed through the successive stages of Swedish language analysis, Swedish-to-English transfer, and English language generation. Finally, the two sets of English outputs were pairwise compared. Language processing was carried out using a robust fallback mechanism (described elsewhere), so that a translation was always produced.

We have noticed when testing and demonstrating the SLT system that people give widely different judgments as to whether a translation is "acceptable". Indeed, it seems unlikely to us that this notion can be given a clear definition independent of a specific context of use. We have also observed, however, that there is much greater agreement on the *relative* quality of different translations. Given two candidate translations of the same utterance, it is normally not controversial to claim that one is better than the other, or that they are in practice equally good.

Our experiments involved 444 test utterances, 41 of which gave rise to different translations when compound splitting was introduced. These 41 utterances were exam-

	Split better	Unsplit better	Unclear
Judge 1	19	12	10
Judge 2	24	11	6
Judge 3	19	10	12

Table 16.4: End-to-end evaluation comparison, giving each judge's preferences for utterances where the translation was affected by compound splitting.

ined by three independent judges, who were all native speakers of English and fluent in Swedish. Each utterance was presented together with the two candidate translations produced by the "split" and "unsplit" versions of the system, respectively: each judge was asked to state whether translation 1 was better or worse than translation 2, or alternatively that neither translation was clearly better than the other. The order in which the two translations were presented—that is, "split" before "unsplit" or *vice versa*—was decided randomly in each case. The results are summarized in Table 16.4. Agreement between the judges was good: in only five sentences out of the 41, a pair of judges gave opposite judgments, one marking the split version as better and the other marking it as worse.

It is interesting to note that although the unsplit version was in several cases better than the split one, the errors in the split translations were never actually caused by failure on the part of the CLE (see Section 4.2) to reassemble a split compound.

Language processing for split compounds

Language processing in SLT is carried out by the Core Language Engine (CLE), a general language-processing system, which has been developed by SRI International in a series of projects starting in 1986. The original system was for English only. The Swedish version (Gamback and Rayner, 1992) was developed in a collaboration with the Swedish Institute of Computer Science. The CLE is extensively described elsewhere (Agnäs *et al.*, 1994; Alshawi *et al.*, 1992; Alshawi (ed), 1992) so we only give the minimum background necessary for understanding our handling of compounds.

The basic functionality offered by the CLE is two-way translation between surface form and a representation in terms of a logic-based formalism called Quasi Logical Form (QLF). The modules consistuting a version of the CLE for a given language can be divided into three groups, which we refer to as "code", "rules," and "preferences". The "code" modules constitute the language-independent compilers and interpreters that make up the basic processing engine; the other two types of module between them constitute a declarative description of the language.

The "rules" contain domain-independent lexico-grammatical information for the language in question; they encode a relationship between surface strings and QLF representations. Thus, for any given surface string, the rules define a set of possible QLF representations of that string. Conversely, given a well-formed QLF representation, the rules can be used to produce a set of possible surface-form realizations of the QLF. The

code modules support compilation of the rules into forms that allow fast processing in both directions: surface-form → QLF (analysis) and QLF → surface-form (generation).

The relationship between surface form and QLF is in general many-to-many. "Preference" modules contain data in the form of statistically learned distributional facts, based on analysis of domain corpora (Alshawi and Carter, 1994). Using this extra information, the system can distinguish between plausible and implausible applications of the rules with a fairly high degree of accuracy.

The principled grammar-based architecture of the CLE made it simple to modify the speech-language interface (Carter and Rayner, 1994) to accommodate input in split-compound format. Since morphology and syntax rules have the same form (Alshawi *et al.*, 1992, Section 3.9) all that was necessary was to change the status of compounding rules from "morphology" to "syntax". In a little more detail:

- Declarations were supplied to identify some morphology rules as specifically compounding rules.
- A switch was added, which, when On, allowed the designated morphology rules to be used as syntax rules.

After a little experimentation, it also turned out to be advantageous to add a few dozen lexicon entries, to cover words that could potentially be constructed as compounds, but in reality are noncompounds. These were automatically generated from the lexicon by using a simple algorithm. No other changes to the system were made, and the adaptation process required only two person-days of work.

16.3.5 Conclusions

To summarize the results of our experiments on ATIS-S-1, we found that compound splitting introduced an undramatic but tangible improvement in both WER and end-to-end system performance. It decreased the vocabulary size for both speech and language processing, and required no substantial modification of any part of the system. Our overall conclusion is that it is a clear win.

We were nonetheless somewhat disappointed to find that the improvement resulting from compound splitting was not larger. We believe that one reason for this lack of improvement was the very small number of translators used to create ATIS-S-1, which led to an unnaturally uniform and homogeneous corpus; in particular, the OOV rate on the test portion, even without compound splitting, is only about 0.5 %. Preliminary results on a new version of Swedish ATIS, ATIS-S-2, support this hypothesis.

ATIS-S-2 has been created in roughly the same way as ATIS-S-1, but using a much larger number of translators, 427 in all. The result, a text corpus containing 4 592 sentences, is a considerably more reasonable approximation to a "real" Swedish ATIS corpus. We merged the ATIS-S-1 and ATIS-S-2 corpora, taking half of ATIS-S-2 as test data and the remaining material as training. Examining this new data, about 5 % of all tokens in the test set are compounds, and the OOV rate of the full test set is 3.0 %. In contrast, the OOV rate measured just on compounds is nearly 23 %. However, if

compounds are split, OOV falls from 3.0 % to 2.1 % (30 % relative) on the whole test-set, and from 23 % to 7 % (70 % relative) on compounds only. The above statistics give us reason to expect that the effect of compound-splitting on WER and end-to-end performance would be rather greater on a more realistic corpus.

Chapter 17

Summary and Conclusions

David Carter, Jaan Kaja, Leonardo Neumeyer and Manny Rayner

This chapter will be completed for the final version of the report. It will describe global system performance.

Appendix A

Morphology

David Carter

In this appendix, we describe the CLE's compiler and development environment for feature-augmented two-level morphology rules. This part of the CLE is used in the SLT system but the work involved was done under other funding.

The compiler is optimized for a class of languages including many or most European ones, and for rapid development and debugging of descriptions of new languages. The key design decision is to compose morphophonological and morphosyntactic information, but not the lexicon, when compiling the description. This results in typical compilation times of about a minute, and has allowed a reasonably full, feature-based description of French inflectional morphology to be developed in about a month by a linguist new to the system.

A.1 Introduction

The paradigm of two-level morphology (Koskenniemi, 1983) has become popular for handling word formation phenomena in a variety of languages. The original formulation has been extended to allow morphotactic constraints to be expressed by feature specification (Trost, 1990; Alshawi *et al*, 1991) rather than Koskenniemi's less perspicuous device of continuation classes. Methods for the automatic compilation of rules from a notation convenient for the rule-writer into finite-state automata have also been developed, allowing the efficient analysis and synthesis of word forms. The automata may be derived from the rules alone (Trost, 1990), or involve composition with the lexicon (Karttunen, Kaplan and Zaenen, 1992).

However, there is often a trade-off between run-time efficiency and factors important for rapid and accurate system development, such as perspicuity of notation, ease of debugging, speed of compilation and the size of its output, and the independence of the morphological and lexical components. In compilation, one may compose any or all of

(a) the two-level rule set,

(b) the set of affixes and their allowed combinations, and

(c) the lexicon;

see Kaplan and Kay (1994) for an exposition of the mathematical basis. The type of compilation appropriate for rapid development and acceptable run-time performance depends on, at least, the nature of the language being described and the number of base forms in the lexicon; that is, on the position in the three-dimensional space defined by (a), (b) and (c).

For example, English inflectional morphology is relatively simple; dimensions (a) and (b) are fairly small, so if (c), the lexicon, is known in advance and is of manageable size, then the entire task of morphological analysis can be carried out at compile time, producing a list of analysed word forms which need only be looked up at run time, or a network which can be traversed very simply. Alternatively, there may be no need to provide as powerful a mechanism as two-level morphology at all; a simpler device such as affix stripping (Alshawi, 1992, p119ff) or merely listing all inflected forms explicitly may be preferable.

For agglutinative languages such as Korean, Finnish and Turkish (Kwon and Karttunen, 1994; Koskenniemi, 1983; Oflazer, 1993), dimension (b) is very large, so creating an exhaustive word list is out of the question unless the lexicon is trivial. Compilation to a network may still make sense, however, and because these languages tend to exhibit few non-concatenative morphophonological phenomena other than vowel harmony, the continuation class mechanism may suffice to describe the allowed affix sequences at the surface level.

Many European languages are of the inflecting type, and occupy still another region of the space of difficulty. They are too complex morphologically to yield easily to the simpler techniques that can work for English. The phonological or orthographic changes involved in affixation may be quite complex, so dimension (a) can be large, and a feature mechanism may be needed to handle such varied but interrelated morphosyntactic phenomena as umlaut (Trost, 1991), case, number, gender, and different morphological paradigms. On the other hand, while there may be many different affixes, their possibilities for combination within a word are fairly limited, so dimension (b) is quite manageable.

This appendix describes a representation and associated compiler intended for two-level morphological descriptions of the written forms of inflecting languages. The CLE supports both a built-in lexicon and access to large external lexical databases, and in that context, highly efficient word analysis and generation at run-time are less important than ensuring that the morphology mechanism is expressive, is easy to debug, and allows relatively quick compilation. Morphology also needs to be well integrated with other processing levels. In particular, it should be possible to specify relations among morphosyntactic and morphophonological rules and lexical entries; for the convenience of developers, this is done by means of feature equations. Further, it cannot be assumed that the lexicon has been fully specified when the morphology rules are compiled. Developers may wish to add and test further lexical entries without frequently recompiling the rules, and it may also be necessary to deal with unknown words at run time, for example by querying a large external lexical database or attempting spelling correction

(Alshawi, 1992, pp124-7). Also, both analysis and generation of word forms are required. Run-time speed need only be enough to make the time spent on morphology small compared to sentential and contextual processing.

These parameters – languages with a complex morphology/syntax interface but a limited number of affix combinations, tasks where the lexicon is not necessarily known at compile time, bidirectional processing, and the need to ease development rather than optimize run-time efficiency – dictate the design of the morphology compiler described in this appendix, in which spelling rules and possible affix combinations (items (a) and (b)), but not the lexicon (item (c)), are composed in the compilation phase. Descriptions of French, Polish and English inflectional morphology have been developed for it, and I show how various aspects of the mechanism allow phenomena in these languages to be handled.

A.2 The Description Language

A.2.1 Morphophonology

The formalism for *spelling rules* (dimension (a)) is a syntactic variant of that of Ruessink (1989) and Pulman (1991). A rule is of the form

$$\texttt{spell(}Name\texttt{, }Surface\ Op\ Lexical\texttt{, }Classes\texttt{, }Features\texttt{).}$$

Rules may be optional (Op is "$\Rightarrow$") or obligatory (Op is "$\Leftrightarrow$"). $Surface$ and $Lexical$ are both strings of the form

$$"LContext|Target|RContext"$$

meaning that the surface and lexical targets may correspond if the left and right contexts and the $Features$ specification are satisfied. The vertical bars simply separate the parts of the string and do not themselves match letters. The correspondence between surface and lexical strings for an entire word is licensed if there is a partitioning of both so that each partition (pair of corresponding surface and lexical targets) is licensed by a rule, and no partition breaks an obligatory rule. A partition breaks an obligatory rule if the surface target does not match but everything else, including the feature specification, does.

The $Features$ in a rule is a list of $Feature = Value$ equations. The allowed (finite) set of values of each feature must be prespecified. $Value$ may be atomic or it may be a boolean expression.

Members of the surface and lexical strings may be characters or classes of single characters. The latter are represented by a single digit N in the string and an item $N/ClassName$ in the $Classes$ list; multiple occurrences of the same N in a single rule must all match the same character in a given application.

Figure A.1 shows three of the French spelling rules developed for this system. The `change_e_è1` rule (simplified slightly here) makes it obligatory for a lexical *e* to be realised as a surface *è* when followed by t, r, or l, then a morpheme boundary, then e, as long as the feature `cdouble` has an appropriate value. The `default` rule that copies characters between surface and lexical levels and the `boundary` rule that

```
spell(change_e_è1, "|è|" ⇔ "|e|1+e",
          [1/trl], [cdouble=n]).
spell(default, "|1|" ⇒ "|1|", [1/letter], []).
spell(boundary, "||" ⇒ "|1|", [1/bmarker], []).
```

Figure A.1: Three spelling rules

Surface:	c	h	è	r		e	
Lexical:	c	h	e	r	+	e	+
Rule:	$def.$	$def.$	*c.e_è1*	$def.$	$bdy.$	$def.$	$bdy.$

Figure A.2: Partitioning of *chère* as `cher+e+`

deletes boundary markers are both optional. Together these rules permit the following realization of *cher* ("expensive") followed by *e* (feminine gender suffix) as *chère*, as shown in Figure A.2. Because of the obligatory nature of `change_e_è1`, and the fact that the orthographic feature restriction on the root *cher*, `[cdouble=n]`, is consistent with the one on that rule, an alternative realisation *chere*, involving the use of the `default` rule in third position, is ruled out.[1]

Unlike many other flavours of two-level morphology, the $Target$ parts of a rule need not consist of a single character (or class occurrence); they can contain more than one, and the surface target may be empty. This obviates the need for "null" characters at the surface. However, although surface targets of any length can usefully be specified, it is in practice a good strategy always to make lexical targets exactly one character long, because, by definition, an obligatory rule cannot block the application of another rule if their lexical targets are of different lengths. The example in Section A.4.1 below clarifies this point.

A.2.2 Word Formation and Interfacing to Syntax

The allowed sequences of morphemes, and the syntactic and semantic properties of morphemes and of the words derived by combining them, are specified by morphosyntactic *production rules* (dimension (b)) and lexical entries both for affixes (dimension (b)) and for roots (dimension (c)), essentially as described by Alshawi (1992) (where the production rules are referred to as "morphology rules"). Affixes may appear explicitly in production rules or, like roots, they may be assigned complex feature-valued categories. Information, including the creation of logical forms, is passed between constituents in a rule by the sharing of variables. These feature-augmented production rules are just the same device as those used in the CLE's syntactico-semantic descriptions, and are a much more natural way to express morphotactic information than finite-state devices such as continuation classes (see Trost and Matiasek, 1994, for a related approach).

[1]The `cdouble` feature is in fact used to specify the spelling changes when *e* is added to various stems: `cher+e`=*chère*, `achet+e`=*achète*, but `jet+e`=*jette*.

```
morph(adjp_adjp_fem, % rule (syntax)
   % mother category:
  [adjp:[agr= @agr(3,sing,f) | Shared],
   % first daughter (category):
   adjp:[agr= @agr(3,sing,m) | Shared],
   % second daughter (literal)
   e])
   % shared syntactic features:
    :- Shared=[aform=Aform, ..., wh=n].

deriv(adjp_adjp_fem, only, % rule (semantics)
   % mother logical form and category:
  [(Adj,adjp:Shared),
   % first daughter:
   (Adj,adjp:Shared),
   % second daughter:
   (_,e)])
   % shared semantic features:
    :- Shared=[anaIn=Ai, ..., subjval=Subj].
```

Figure A.3: Syntactic and semantic morphological production rules

The syntactic and semantic production rules for deriving the feminine singular of a French adjective by suffixation with "`e`" are given, with some details omitted, in Figure A.3. In this case, nearly all features are shared between the inflected word and the root, as is the logical form for the word (shown as `Adj` in the `deriv` rule). The only differing feature is that for gender, shown as the third argument of the `@agr` macro, which itself expands to a category.

Irregular forms, either complete words or affixable stems, are specified by listing the production rules and terminal morphemes from which the appropriate analyses may be constructed, for example:

```
irreg(dit,[dire,'PRESENT_3s'],[v_v_affix-only]).
```

Here, `PRESENT_3s` is a pseudo-affix which has the same syntactic and semantic information attached to it as (one sense of) the affix "`t`", which is used to form some regular third person singulars. However, the spelling rules make no reference to `PRESENT_3s`; it is simply a device allowing categories and logical forms for irregular words to be built up using the same production rules as for regular words.

A.3 Compilation

All rules and lexical entries in the CLE are compiled to a form that allows normal Prolog unification to be used for category matching at run time. The same compiled forms are used for analysis and generation, but are indexed differently. Each feature

for a major category is assigned a unique position in the compiled Prolog term, and features for which finite value sets have been specified are compiled into vectors in a form that allows boolean expressions, involving negation as well as conjunction and disjunction, to be conjoined by unification (see Mellish, 1988; Alshawi, 1992, pp46–48).

The compilation of morphological information is motivated by the nature of the task and of the languages to be handled. As discussed in Section A.1, we expect the number of affix combinations to be limited, but the lexicon is not necessarily known in advance. Morphophonological interactions may be quite complex, and the purpose of morphological processing is to derive syntactic and semantic analyses from words and vice versa for the purpose of full NLP. Reasonably quick compilation is required, and run-time speed need only be moderate.

A.3.1 Compiling Spelling Patterns

Compilation of individual `spell` rules is straightforward; feature specifications are compiled to positional/boolean format, characters and occurrences of character classes are also converted to boolean vectors, and left contexts are reversed (cf Abramson, 1992) for efficiency. However, although it would be possible to analyse words directly with individually compiled rules (see Section A.5 below), it can take an unacceptably long time to do so, largely because of the wide range of choices of rule available at each point and the need to check at each stage that obligatory rules have not been broken. We therefore take the following approach.

First, all legal sequences of morphemes are produced by top-down nondeterministic application of the production rules (Section A.2.2), selecting affixes but keeping the root morpheme unspecified because, as explained above, the lexicon is undetermined at this stage. For example, for English, the sequences `*+ed+ly` and `un+*+ing` are among those produced, the asterisk representing the unspecified root.

Then, each sequence, together with any associated restrictions on orthographic features, undergoes analysis by the compiled spelling rules (Section A.2.1), with the surface sequence and the root part of the lexical sequence initially uninstantiated. Rules are applied recursively and nondeterministically, somewhat in the style of Abramson (1992), taking advantage of Prolog's unification mechanism to instantiate the part of the surface string corresponding to affixes and to place some spelling constraints on the start and/or end of the surface and/or lexical forms of the root.

This process results in a set of *spelling patterns*, one for each distinct application of the spelling rules to each affix sequence suggested by the production rules. A spelling pattern consists of partially specified surface and lexical root character sequences, fully specified surface and lexical affix sequences, orthographic feature constraints associated with the spelling rules and affixes used, and a pair of syntactic category specifications derived from the production rules used. One category is for the root form, and one for the inflected form.

Spelling patterns are indexed according to the surface (for analysis) and lexical (for generation) affix characters they involve. At run time, an inflected word is analysed nondeterministically in several stages, each of which may succeed any number of times including zero:

- stripping off possible (surface) affix characters in the word and locating a spelling pattern that they index;
- matching the remaining characters in the word against the surface part of the spelling pattern, thereby, through shared variables, instantiating the characters for the lexical part to provide a possible root spelling;
- checking any orthographic feature constraints on that root;
- finding a lexical entry for the root, by any of a range of mechanisms including lookup in the system's own lexicon, querying an external lexical database, or attempting to guess an entry for an undefined word; and
- unifying the root lexical entry with the root category in the spelling pattern, thereby, through variable sharing with the other category in the pattern, creating a fully specified category for the inflected form that can be used in parsing.

In generation, the process works in reverse, starting from indexes on the lexical affix characters.

A.3.2 Representing Lexical Roots

Complications arise in spelling rule application from the fact that, at compile time, neither the lexical nor the surface form of the root, nor even its length, is known. It would be possible to hypothesize all sensible lengths and compile separate spelling patterns for each. However, this would lead to many times more patterns being produced than are really necessary.

Lexical (and, after instantiation, surface) strings for the unspecified roots are therefore represented in a more complex but less redundant way: as a structure

$$L_1 \ldots L_m \texttt{v(}L, R\texttt{)}\ R_1 \ldots R_n.$$

Here the L_i's are variables later instantiated to single characters at the beginning of the root, and L is a variable, which is later instantiated to a list of characters, for its continuation. Similarly, the R_i's represent the end of the root, and R is the continuation (this time reversed) leftwards into the root from R_1. The `v(`L, R`)` structure is always matched specially with a Kleene-star of the `default` spelling rule. For full generality and minimal redundancy, L_m and R_1 are constrained not to match the default rule, but the other L_i's and R_i's may. The values of n required are those for which, for some spelling rule, there are k characters in the target lexical string and $n-k$ from the beginning of the right context up to (but not including) a boundary symbol. The lexical string of that rule may then match $R_1, \ldots, R_k$, and its right context match $R_{k+1}, \ldots, R_n$,`+`, ...
The required values of m may be calculated similarly with reference to the left contexts of rules.[2]

[2] Alternations in the middle of a root, such as umlaut, can be handled straightforwardly by altering the root/affix pattern from $L_1 \ldots L_m$ `v(`L, R`)` $R_1 \ldots R_n$ to $L_1 \ldots L_m$ `v(`L, R`)` M v(L', R') $R_1 \ldots R_n$, with M forbidden to be the `default` rule. This has not been necessary for the descriptions developed so far, but its implementation is not expected to lead to any great decrease in run-time performance, because the non-determinism it induces in the lookup process is no different in kind from that arising from alternations at root-affix boundaries.

Compile time:	Rule:	$def.^*$		*c.e_è1*	$def.$	$bdy.$	$def.$	$bdy.$
	Variable:	v(L, R)		R_1	R_2	...		
Run time:	Surface:	c	h	è	r		e	
	Lexical:	c	h	e	r	+	e	+

Figure A.4: Spelling pattern application to the analysis of *chère*

During rule compilation, the spelling pattern that leads to the run-time analysis of *chère* given above is derived from $m = 0$ and $n = 2$ and the specified rule sequence, with the variables R_1 R_2 matching as in Figure A.4.

A.3.3 Applying Obligatory Rules

In the absence of a lexical string for the root, the correct treatment of obligatory rules is another problem for compilation. If an obligatory rule specifies that lexical X must be realised as surface Y when certain contextual and feature conditions hold, then a partitioning where X is realised as something other than Y is only allowed if one or more of those conditions is unsatisfied. Because of the use of boolean vectors for both features and characters, it is quite possible to constrain each partitioning by unifying it with the complement of one of the conditions of each applicable obligatory rule, thereby preventing that rule from applying. For English, with its relatively simple inflectional spelling changes, this works well. However, for other languages, including French, it leads to excessive numbers of spelling patterns, because there are many obligatory rules with non-trivial contexts and feature specifications.

For this reason, complement unification is not actually carried out at compile time. Instead, the spelling patterns are augmented with the fact that certain conditions on certain obligatory rules need to be checked on certain parts of the partitioning when it is fully instantiated. This slows down run-time performance a little but, as we will see below, the speed is still quite acceptable.

A.3.4 Timings

The compilation process for the entire rule set takes just over a minute for a fairly thorough description of French inflectional morphology, running on a Sparcstation 10/41 (SPECint92=52.6). Run-time speeds are quite adequate for full NLP, and reflect the fact that the system is implemented in Prolog rather than (say) C and that full syntactico-semantic analyses of sentences, rather than just morpheme sequences or acceptability judgments, are produced.

Analysis of French words using this rule set and only an in-core lexicon averages around 50 words per second, with a mean of 11 spelling analyses per word leading to a mean of 1.6 morphological analyses (the reduction being because many of the roots suggested by spelling analysis do not exist or cannot combine with the affixes produced). If results are cached, subsequent attempts to analyse the same word are around 40 times faster still. Generation is also quite acceptably fast, running at around

Surface:	b	e	a	u		e	
Lexical:	b	e	a	u	+	e	+
Rule:	$def.$	$def.$	$def.$	$def.$	$bdy.$	$def.$	$bdy.$

Figure A.5: Incorrect partitioning for `beau+e+`

100 words per second; it is slightly faster than analysis because only one spelling, rather than all possible analyses, is sought from each call. Because of the separation between lexical and morphological representations, these timings are essentially unaffected by in-core lexicon size, as full advantage is taken of Prolog's built-in indexing.

Development times are at least as important as computation times. A rule set embodying a quite comprehensive treatment of French inflectional morphology was developed in about one person month. The English spelling rule set was adapted from Ritchie *et al* (1992) in only a day or two. A Polish rule set is also under development, and Swedish is planned for the near future.

A.4 Some Examples

To clarify further the use of the formalism and the operation of the mechanisms, we now examine several further examples.

A.4.1 Multiple-letter spelling changes

Some obligatory spelling changes in French involve more than one letter. For example, masculine adjectives and nouns ending in *eau* have feminine counterparts ending in *elle*: *beau* ("nice") becomes *belle*, *chameau* ("camel") becomes *chamelle*. The final *e* is a feminizing affix and can be seen as inducing the obligatory spelling change *au* $\rightarrow$ *ll*. However, although the obvious spelling rule,

```
spell(change_au_ll, "|ll|" ↔ "|au|+e"),
```

allows this change, it does not rule out the incorrect realization of `beau+e` as **beaue*, shown in Figure A.5, because it only affects partitionings where the `au` at the lexical level forms a *single* partition, rather than one for `a` and one for `u`. Instead, the following pair of rules, in which the lexical targets have only one character each, achieve the desired effect:

```
spell(change_au_ll1, "|l|" ↔ "|a|u+e")
spell(change_au_ll2, "|l|" ↔ "a|u|+e")
```

Here, `change_au_ll1` rules out the `a:a` partition in Figure A.5, and `change_au_ll2` rules out the `u:u` one.

It is not necessary for the *surface* target to contain exactly one character for the blocking effect to apply, because the semantics of obligatoriness is that the *lexical* target and all contexts, taken together, make the specified *surface* target (of whatever

Surface:	b	o	j		e	
Lexical:	b	ó	j	+	e	+
Rule:	$def.$	*c_ó_o.*	$def.$	$bdy.$	$def.$	$bdy.$

Surface:	z	b	ó	j		e	
Lexical:	z	b	ó	j	+	e	+
Rule:	$def.$	$def.$	$def.$	$def.$	$bdy.$	$def.$	$bdy.$

Figure A.6: Feature-dependent dropping of accent

length) obligatory for that partition. The reverse constraint, on the lexical target, does not apply.

A.4.2 Using features to control rule application

Features can be used to control the application of rules to particular lexical items where the applicability cannot be deduced from spellings alone. For example, Polish nouns with stems whose final syllable has vowel *ó* normally have inflected forms in which the accent is dropped. Thus in the nominative plural, *krój* ("style") becomes *kroje*, *bór* ("forest") becomes *bory*, *bój* ("combat") becomes *boje*. However, there are exceptions, such as *zbój* ("bandit") becoming *zbóje*. Similarly, some French verbs whose infinitives end in *-eler* take a grave accent on the first *e* in the third person singular future (*modeler*, "model", becomes *modèlera*), while others double the *l* instead (e.g. *appeler*, "call", becomes *appellera*).

These phenomena can be handled by providing an obligatory rule for the case whether the letter changes, but constraining the applicability of the rule with a feature and making the feature clash with that for roots where the change does not occur. In the Polish case:

```
spell(change_ó_o, "|o|" ↔ "|ó|1+2",
            [1/c, 2/v], [chngo=y]).

orth(zbój, [chngo=n]).
```

Then the partitionings given in Figure A.6 will be the only possible ones. For *bój*, the `change_ó_o` rule must apply, because the `chngo` feature for *bój* is unspecified and therefore can take any value; for *zbój*, however, the rule is prevented from applying by the feature clash, and so the default rule is the only one that can apply.

A.5 Debugging the Rules

The debugging tools help in checking the operation of the spelling rules, either (1) in conjunction with other constraints or (2) on their own.

For case (1), the user may ask to see all inflections of a root licensed by the spelling rules, production rules, and lexicon; for *cher*, the output is

```
"chère" has root "cher" with pattern 194 and tree
17.

Pattern 194:

    "___è{clmnprstv=A}e" <-> "___e{clmnprstv=A}+e+"
    => tree 17 and 18 if [doublec=n]
    Uses: default* change_e_è1 default boundary
          default boundary


Tree 17:

   Both = adjp:[dmodified=n,headfinal=y,mhdfl=y,
                synmorpha=1,wh=n]
   Root = adjp:[agr=agr:[gender=m]]
   Infl = adjp:[agr=agr:[gender=f]]
   Tree = adjp_adjp_fem=>[*,e]
```

Figure A.7: Debugger trace of derivation of *chère*

```
[cher,e]: adjp -> chère
[cher,e,s]: adjp -> chères
[cher,s]: adjp -> chers
```

meaning that when *cher* is an `adjp` (adjective) it may combine with the suffixes listed to produce the inflected forms shown. This is useful in checking over- and undergeneration. It is also possible to view the spelling patterns and production rule tree used to produce a form; for *chère*, the trace (slightly simplified here) is as in Figure A.7. The spelling pattern 194 referred to here is the one depicted in a different form in Figure A.4. The notation `{clmnprstv=A}` denotes a set of possible consonants represented by the variable `A`, which also occurs on the right hand side of the rule, indicating that the same selection must be made for both occurrences. Production rule tree 17 is that for a single application of the rule `adjp_adjp_fem`, which describes the feminine form of the an adjective, where the root is taken to be the masculine form. The `Root` and `Infl` lines show the features that differ between the root and inflected forms, while the `Both` line shows those that they share. Tree 18, which is also pointed to by the spelling pattern, describes the feminine forms of nouns analogously.

For case (2), the spelling rules may be applied directly, just as in rule compilation, to a specified surface or lexical character sequence, as if no lexical or morphotactic constraints existed. Feature constraints, and cases where the rules will not apply if those constraints are broken, are shown. For the lexical sequence `cher+e+`, for example, the output is as follows.

```
Surface: "chère" <->
Lexical: "cher". Suffix: "e"
```

```
c :: c <- default
h :: h <- default
è :: e <- change_e_è1
r :: r <- default
  :: + <- boundary
Category: orth:[cdouble=n]
e :: e <- default
  :: + <- boundary


Surface: "chere" <->
Lexical: "cher". Suffix: "e"
c :: c <- default
h :: h <- default
e :: e <- default  (breaks "change_e_è1")
r :: r <- default
  :: + <- boundary
e :: e <- default
  :: + <- boundary
```

This indicates to the user that if *cher* is given a lexical entry consistent with the constraint `cdouble=n`, then only the first analysis will be valid; otherwise, only the second will be.

A.6 Conclusions and Further Work

The rule formalism and compiler described here work well for European languages with reasonably complex orthographic changes but a limited range of possible affix combinations. Development, compilation and run-time efficiency are quite acceptable, and the use of rules containing complex feature-augmented categories allows morphotactic behaviours and non-segmental spelling constraints to be specified in a way that is perspicuous to linguists, leading to rapid development of descriptions adequate for full NLP.

The kinds of non-linear effects common in Semitic languages, where vowel and consonant patterns are interpolated in words (Kay, 1987; Kiraz, 1994) could be treated efficiently by the mechanisms described here if it proved possible to define a representation that allowed the parts of an inflected word corresponding to the root to be separated fairly cleanly from the parts expressing the inflection. The latter could then be used by a modified version of the current system as the basis for efficient lookup of spelling patterns which, as in the current system, would allow possible lexical roots to be calculated.

Agglutinative languages could be handled efficiently by the current mechanism if specifications were provided for the affix combinations that were likely to occur at all often in real texts. A backup mechanism could then be provided which attempted a slower, but more complete, direct application of the rules for the rarer cases.

The interaction of morphological analysis with spelling correction (Carter, 1992; Oflazer, 1994; Bowden, 1995) is another possibly fruitful area of work. Once the root spelling patterns and the affix combinations pointing to them have been created, analysis essentially reduces to an instance of affix-stripping, which would be amenable to exactly the technique outlined by Carter (1992). As in that work, a discrimination net of root forms would be required; however, this could be augmented independently of spelling pattern creation, so that the flexibility resulting from not composing the lexicon with the spelling rules would not be lost.

Appendix B

SLT in the Media

Robert Eklund

In this appendix, media coverage of the SLT projects will be summarised. Since they are leading-edge research, the two SLT projects have received a good deal of attention from the media. Moreover, media interest increased during the SLT-2 phase, especially in Sweden, probably due to the existence of a state-of-the-art Swedish speech recognizer. This appendix will list and summarise the media coverage the two SLT projects have received. The contents of the articles will be briefly accounted for, as will the particular media in which they appeared and some indication of circulation figures where available.

B.1 SLT-1

During SLT-1, media interest was sparse, with a few, though notable, exceptions. Still, nationwide exposure was catered for through the article in Aftonbladet (Section B.1.1 below), and at least all Telia employees were made aware of the activities through the article in Televärlden (Section B.1.3). The article in Computer Sweden (Section B.1.2) also introduced computer-literate Swedes to the SLT system.

B.1.1 Aftonbladet

Aftonbladet is one of Sweden's biggest daily newspapers. At the time of the article, it was the second biggest, with a circulation of about 400,000.

Issue ??

Article Magnus Ringman: *Nu vet Bildt hur framtiden ser ut* (Now Bildt knows what the future looks like), p. ?

Content Swedish premier minister Carl Bildt – known to be an avid proponent of IT – paid a visit to the Swedish Institute of Computer Science (SICS), an SLT party at the time. The English-to-Swedish translation was demonstrated to him, and

he was given the opportunity to try it out himself. Christer Samuelsson quoted. Picture of Carl Bildt in front of a Sun Sparcstation, wearing a headset, while addressing the SLT application.

B.1.2 Computer Sweden

Computer Sweden is a medium covering the computer world.

Issue Friday 17 December 1993.

Article Tomas Zirn: *Låt datorn sköta översättningen* (Let the computer do the translation). p. 31.

Content This article is a fairly detailed description of the English-to-Swedish version of the Spoken Language Translator. Telia Research AB, SICS and SRI International are mentioned by name. Interviews with Björn Gambäck. Picture of Ivan Bretan in front of SUN Sparc station with headset.

B.1.3 Televärlden (1)

Televärlden is Telia's official internal medium, distributed to all Telia employees.

Issue No. 17, 1993.

Article Jens Busch: *Engelskt tal blir svenskt – dator översätter muntliga flygbokningar* (English speech becomes Swedish – computer translates oral flight bookings), p. ??.

Content This article constitutes a small presentation of SLT-1. The project is not mentioned by name, but Telia Research AB, SRI International and SICS are. Interview with Bertil Lyberg, Telia Research. Stephen Pulman of SRI, Cambridge, is mentioned.

B.1.4 Televärlden (2)

See Section B.1.3 for a general description of the medium.

Issue [BERTIL TO CHECK]

Article Jens Busch: *Telefonen skall bli översättare* (The telephone will be a translator.), p. ??.

Content This article discusses possible applications based on speech technology. Interview with Bertil Lyberg. SLT parties SRI International and SICS are mentioned by name.

B.2 SLT-2

During SLT-2, media interest increased, resulting in wide exposure both geographically and professionally. In the following paragraphs, the articles and TV features are listed in chronological order.

B.2.1 Verkstäderna

Verkstäderna – Tidskriften för Sveriges Verkstadsindustrier is a technical magazine published by Verkstädernas Förlag AB. It has a membership circulation to eight industrial organizations.

Issue No. 11, 1995.

Article Eva Regårdh: *Det mänskliga gränssnittet – Tala med datorn* (The Human Interface – Speak with the computer), pp. 6–8.

Content A fairly detailed description of the SLT project. Also a general introduction to speech technology. Interviews with Robert Eklund, Mats Ljungqvist and Bertil Lyberg, all of whom at Telia Research AB.

B.2.2 Mitt i Haninge

Mitt i Haninge is a weekly newspaper distributed to all households in Haninge Kommun (Haninge municipality), where Telia Research is located. The circulation is 30,300.

Issue No. 325, 15th October, 1996, week 42, year 8.

Article Lena Granström: *Datorn ger svar på tal* (The computer talks back).

Content Since Telia is the biggest employer in Haninge Kommun, Mitt i Haninge published a series of articles to present Telia and its subcompanies to the inhabitants of the municipality. Interview with Robert Eklund. The SLT activity (albeit not by name) is described, as are possible future applications based on speech technology.

B.2.3 Expressen

Expressen was at the time of the article Sweden's biggest newspaper, with a circulation of around 400,000. The article was published in the Sunday supplement "exxet".

Issue October 22, 1995.

Article Per Runhammar: *Låt datorn sköta snacket* (Let the computer do the talking), pp. 58–59.

Content This article may be described as a "crash course" description of the SLT project. The project is not mentioned by name, but its parties are. It contains "nutshell" explanations of speech recognition, automatic text translation and speech synthesis. Some examples of possible future applications are mentioned.

B.2.4 BBC World Service

"New Ideas" radio program broadcast on September 1st, 1996. The program emphasized the English-to-French translation, and included interviews with Manny Rayner and David Carter.

B.2.5 Ny Teknik

Ny Teknik – Teknisk Tidsskrift is a technical journal distributed to all Swedish engineers.

Issue Nr 37, 12 september 1996.

Article Håkan Borgström: *Åsa lär datorn förstå svenska* (Åsa teaches the computer to understand Swedish), pp. 12–14. Front page picture of Robert Eklund supervising recording session titled *Datorn förstår hans språk* (The computer understands his language).

Content A synoptic overview of speech technology in Sweden (other institutes are also covered). Interviews with Robert Eklund, Bertil Lyberg and Per Sautermeister. SRI International is mentioned. Picture of Åsa Hällgren of Telia Research in an anechoic chamber. Picture of Per Sautermeister in front of a Sun Sparc station. Figure explaining concatenation synthesis principles.

B.2.6 Metro

Metro is a weekday newspaper distributed free in the public transportation network in Stockholm and suburbs. The circulation is around 237,000.

Issue Wednesday 18th September 1996.

Article Håkan Borgström: *Datorerna som kan stockholmska – forskare på Telia lär datorer att förstå mänskligt tal*. (The computers that understand Stockholmian – Researchers at Telia teaches computers to understand human speech), pp. 12–13. Front page picture (same picture as the one published in Ny Teknik) of Åsa Hällgren in an anechoic chamber titled *Nu lär sig datorn att höra* (Now the computer is learning to hear).

Content This article, which was the cover story here, is basically a rehash of the article published in Ny Teknik. Compared to the version in Ny Teknik, it is adapted for a wider audience. Interviews with Robert Eklund, Bertil Lyberg and Per Sautermeister. Picture of Robert Eklund, Bertil Lyberg and Per Sautermeister on the roof of Telia Research, Haninge. Picture of Robert Eklund supervising recording session.

B.2.7 Televärlden (3)

See Section B.1.3 for description.

Issue No. 12, 27 June 1996.

Article Margareta Johansson: *Han lär datorn prata* (He teaches the computer to speak), p. 19.

Content This article is basically an interview with Robert Eklund of Telia Research AB. It may be described as a general presentation of phonetics, linguistics and speech technology. SLT activities are used as examples but neither the project nor its parties are mentioned by name.

B.2.8 Radio-Vian

Radio-Vian is an internal newspaper published by Telia MobiTel AB.

Issue No. 6, September 1995.

Article Jan Sjöberg: *Kunglig glans över huvudkontoret* (Royal splendour over the headquarter).

Content As a part of the Royal visit to Haninge Kommun in August 1995, the Swedish King and Queen visited Telia. Telia CEO Lars Berg hosted the Telia visit. Telia MobiTel was presented by its CEO Seth Myhrby. Telia Research was presented by its CEO Östen Mäkitalo. As a part of the latter presentation Robert Eklund gave a short demonstration of the English-to-Swedish SLT-1 system. The demonstration is mentioned in the article in very positive terms.

B.2.9 Rapport

Rapport is a major TV network news magazine. It is currently the largest audience in Sweden.

"Issue" December 1996.

Content A very short presentation of the system. A TV reporter tries two utterances. No interviews.

B.2.10 Nova

Nova is a major network, popular scientific TV magazine.

Issue Loosely scheduled for broadcasting in May 1997.

Article Interviews with Rolf Hulthén and Bertil Lyberg of Telia Research AB. Per Sautermeister exemplifies two sentences of the Swedish-to-English translator.

B.3 Final Remarks

Summing up, it may be said that the SLT project has received very good exposure. Thanks to the article in Ny Teknik, most – theoretically all – engineers in Sweden should have had SLT brought to their attention. The article in Mitt i Haninge has exposed the entire population of the municipality to SLT activities. The articles in Televärlden has exposed SLT to all Telia employees. The cover story in Metro meant that most people in the greater Stockholm area were given a good introduction to the SLT project and the field of speech technology in general. Finally, following the SLT-1 exposure in Aftonbladet, the “crash course” article in Expressen meant nationwide exposure.

References

Abramson, H. 1992. "A Logic Programming View of Relational Morphology". Proceedings of COLING-92, 850–854.

Agnäs, M-S., and 17 others 1994. *Spoken Language Translator: First Year Report*. Joint report by SRI International (Cambridge) and SICS.*[1]

Agnäs, M-S., H. Alshawi, I. Bretan, D.M. Carter, K. Ceder, M. Collins, R. Crouch, V. Digalakis, B. Ekholm, B. Gambäck, J. Kaja, J. Karlgren, B. Lyberg, P. Price, S.G. Pulman, M. Rayner, C. Samuelsson, and T. Svensson. 1994. *Spoken Language Translator: First Year Report.**

Alshawi, H. (ed) 1992. *The Core Language Engine*. Cambridge, Massachusetts: MIT Press.

Alshawi, H., D.J. Arnold, R. Backofen, D.M. Carter, J. Lindop, K. Netter, S.G. Pulman, J. Tsujii, and H. Uszkoreit 1991. *Eurotra ET6/1: Rule Formalism and Virtual Machine Design Study*. Commission of the European Communities, Luxembourg.

Alshawi, H., C.G. Brown, D.M. Carter, B. Gambäck, S.G. Pulman, and Manny Rayner. 1991. "Bilingual Conversation Interpreter: A Prototype Message Translator. Final Report". *Joint Research Report R91011 and CCSRC-018*, SICS and SRI International, Stockholm, Sweden and Cambridge, England.*

Alshawi, H., and D.M. Carter. 1994 "Training and Scaling Preference Functions for Disambiguation." *Computational Linguistics*, 20:4.

Alshawi, H., D.M. Carter, R. Crouch, S.G. Pulman, M. Rayner, and A. Smith. 1992. "CLARE: A Contextual Reasoning and Cooperative Response Framework for the Core Language Engine" SRI technical report CRC-028.*

Alshawi, H., and R. Crouch 1992. "Monotonic Semantic Interpretation". In *Proceedings of 30th Annual Meeting of the Association for Computational Linguistics*, pp. 32–39, Newark, Delaware.*

Alshawi, H., and R. Crouch. 1992. "Monotonic Semantic Interpretation". Proceedings of 30th ACL.

[1] Starred references are also available from `http://www.cam.sri.com`.

Amalberti, R., N. Carbonell, and P. Falzon. 1993. “User representations of computer systems in human–computer speech interaction”. *Int. J. Man–Machine Studies*, vol. 38, pp. 547–566.

Andry, F., J. Dowding, M. Gawron, and R. Moore. 1994. “A Tool for Collecting Domain Dependent Sortal Constraints From Corpora.” *Proceedings of COLING-94, Kyoto.*

Aubert, X., R. Haeb-Umbach, and H. Ney. 1993. “Continuous Mixture Densities and Linear Discriminant Analysis for Improved Context-Dependent Acoustic Models,” *Proceedings ICASSP*, pp. 648-65.

Bäckström, M., K. Ceder and B. Lyberg. 1989: “Prophon - an Interactive Environment for Text-to-Speech Conversion”. *Proceedings of the European Conference on Speech Communication and Technology*, Vol. 1, pp. 144-147.

Bès, G., and C. Gardent. 1989. “French Order without Order.” Proceedings of 4th European ACL.

Bowden, T. 1995. “Cooperative Error Handling and Shallow Processing”, these proceedings.

Bretan, I., C. Ereback, C. MacDermid, A. Waern. 1995. “Simulation-Based Dialogue Design for Speech-Controlled Telephone Services”. *Proceedings of CHI’95*, Denver.

Bretan, I., R. Eklund, C. MacDermid. 1996. “Approaches to Gathering Realistic Training Data for Speech Translation Systems.” *IEEE Third Workshop Interactive Voice Technology for Telecommunications Applications*, pp. 97–100, September 30 – October 1, 1996. Basking Ridge, New Jersey.

Briscoe, E.J., and J. Carroll 1993. “Generalized Probabilistic LR Parsing of Natural Language (Corpora) with Unification-Based Grammars.” *Computational Linguistics*, 19:1, pp. 25-60.

Brown, P., S.A. Della Pietra, V.J. Della Pietra. 1992. J.C. Lai, and R.L. Mercer, “Class-Based n-gram Models of natural Language,” *Computational Linguistics*, pp. 31-40, Vol. 18 (4).

Brown, K., and E. B. George. 1995. CTIMIT: A speech corpus for the cellular environment with applications to automatic speech recognition. In *ICASSP-95*, pp. 105–108.

Bruce, G., and E. Gårding. 1978. “A Prosodic Typology for Swedish Dialects”. In *Nordic Prosody*, Travaux de L’Institut de Lund. Department of Linguistics, University of Lund.

Carter, D.M. 1989. “Lexical acquisition in the Core Language Engine” European ACL, Manchester.

Carter, D.M. 1992. “Lattice-based Word Identification in CLARE”. Proceedings of ACL-92.

Carter, D.M. 1995. “Rapid Development of Morphological Descriptions for Full Language Processing Systems.” Proceedings of 7th European ACL. Also SRI Technical Report CRC-047

Carter, D.M., J. Kaja, L. Neumeyer, M. Rayner, F. Weng, M. Wiren. 1996. “Handling Compound Nouns in a Swedish Speech-Understanding System”, Processings of ICSLP-96, June 1996.*

D.M. Carter and M. Rayner. 1994. “The Speech-Language Interface in the Spoken Language Translator”. In *Proc. 8th Twente Workshop on Language Technology*, University of Twente, Enschede, the Netherlands.

Chow Y.L., and R. Schwartz. 1990. “The B-Best Algorithm: An efficient procedure for finding Top N Sentence Hypotheses,’ Proceedings of ICASSP.

Church, K. 1988. “A stochastic parts program and noun phrase parser for unrestricted text.” Proceedings of 1st ANLP, Austin, Tx., pp. 136-143.

Crouch, R. 1995 “Ellipsis and Quantification: A Substitutional Approach”. Proceedings of EACL-95, 229-236

Cutting, D., J. Kupiec, J. Pedersen and P. Sibun. 1992. “A Practical Part-of-Speech Tagger” Proceedings of 3rd ANLP, Trento, Italy, pp. 133-140.

Dagan, I., and A. Itai. 1994. “Word Sense Disambiguation Using a Second Language Monolingual Corpus”, *Computational Linguistics* 20:4, pp. 563–596.

DeMarcken, C.G. 1990. “Parsing the LOB Corpus” *Proceedings of 28th ACL, Pittsburgh, Pa., pp. 243-251*

Dempster, A.P., N.M. Laird and D.B. Rubin. 1977. “Maximum Likelihood Estimation from Incomplete Data,” *Journal of the Royal Statistical Society (B)*, Vol. 39, No. 1, pp. 1–38.

DeRose, S. 1988. “Grammatical Category Disambiguation by Statistical Optimization.” *Computational Linguistics* 14, pp. 31-39

Digalakis, V., P. Monaco and H. Murveit. 1996. “Genones: Generalized Mixture Tying in Continuous Hidden Markov Model-Based Speech Recognizers,” *IEEE Transactions Speech and Audio Processing*, pp. 281-289.

Digalakis, V., and L. Neumeyer. 1996. “Speaker Adaptation Using Combined Transformation and Bayesian Methods,” *IEEE Transactions Speech and Audio Processing*, pp. 294-300.

Digalakis V., L. Neumeyer and D. Rtischev. “Speaker Adaptation Using Constrained Reestimation of Gaussian Mixtures,” *IEEE Transactions Speech and Audio Processing*, pp. 357–366, September 1995.

van Eijck, J. and R. Moore. 1992. “Semantic Rules for English.” In Alshawi (ed), 1992.

Ejerhed, E., G. Källgren, O. Wennstedt and M. Åström. 1992. *The Linguistic Annotation System of the Stockholm–Umeå Corpus Project. Description and Guidelines*. Version 4.31, May 22, 1992. University of Umeå.

Eklund, R. and A. Lindström. 1996. “Pronunciation in an internationalized society: a multi-dimensional problem considered”. FONETIK 1996, Swedish Phonetics Conference, Nässlingen, 29–30 May, 1996. *TMH-QPSR/1996*.

ELAN informatique. 1996. *CNETVOX user's manual*. ELAN informatique, 4, rue Jean Rodier, 31400 Toulouse, France.

Elert, C-C. 1994. "Indelning och gränser inom området för den nu talade svenskan – en aktuell dialektografi". In L-E. Edlund: *Kulturgränser – myt eller verklighet*, pp. 215–228. Diabas, University of Umeå.

Engelson, S., and I. Dagan. 1996. "Minimizing Manual Annotation Cost in Supervised Training from Corpora". In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics*, pp. 319-326, Santa Cruz, CA.

Entropic Research Laboratory, Inc. 1995 *Developer Truetalk* Entropic Research Laboratory, Inc., 600 Pennsylvania Ave., SE Suite 202, Washington, DC 20003, USA.

Estival, D. 1990. "Generating French with a Reversible Unification Grammar." Proceedings of 13th COLING.

Fries, S. 1994. " Dialektgränser och kulturgränser". In L-E. Edlund: *Kulturgränser – myt eller verklighet*, pp. 189–198. Diabas, University of Umeå.

Fromkin, V. (ed.). 1980. *Errors in Linguistic Performance: Slips of the Tongue, Ear, Pen and Hand*. New York Academic Press.

Gambäck, B., and M. Rayner. 1992. "The Swedish Core Language Engine". *Proceedings of 3rd Nordic Conference on Text Comprehension in Man and Machine, Linköping, Sweden.* Also SRI Technical Report CRC-025.

Gauvain, J-L., and C-H. Lee. 1994. "Maximum a Posteriori Estimation for Multivariate Gaussian Observations of Markov Chains," *IEEE Transactions Speech and Audio Processing*, pp. 291–298.

Geutner, P. 1995. "Using Morphology Towards Better Large-Vocabulary Speech Recognition Systems". In *Proceedings of ICASSP 95*, 1995.

Grimshaw, J. 1982. "On the Lexical Representation of Romance Reflexives." In J. Bresnan (ed.), The Mental Representation of Grammatical Relations. MIT Press.

Grishman, R., L. Hirschmann, E. Marsh and N. Nhan. 1984. "Automated Determination of Sublanguage Usage." *Proceedings of 22nd COLING, Stanford, pp. 96-100.*

van Harmelen, F., A. Bundy. 1988. "Explanation-Based Generalization = Partial Evaluation" (Research Note) Artificial Intelligence 36, pp. 401–412.

Gårding, E. 1975. "Toward a Prosodic Typology for Swedish Dialects". In K-H. Dahlstedt (ed.): *The Nordic Languages and Modern Linguistics 2*. pp. 466–474. Almqvist & Wiksell, Stockholm.

Hemphill, C.T., J.J. Godfrey and G.R. Doddington. 1990. "The ATIS Spoken Language Systems pilot corpus." Proceedings of DARPA Speech and Natural Language Workshop, Hidden Valley, Pa., pp. 96-101.

Hetherington, I.L., V.W. Zue. 1993. "New Words: Implications for Continuous Speech Recognition". *Proc. Eurospeech 1993*.

Kaplan, R.M., and M. Kay. 1994. "Regular Models of Phonological Rule Systems", *Computational Linguistics*, 20:3, 331–378.

Karlsson, F., A. Anttila, J. Heikkilä and A. Voutilainen (eds). 1995. *Constraint Grammar.* Mouton de Gruyer, Berlin, New York.

Karttunen, L., R.M. Kaplan, and A. Zaenen. 1992. "Two-level Morphology with Composition". *Proceedings of COLING-92, 141–148.*

Katz, S.M. 1987. "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Trans. on Acoust., Speech and Signal Proc., ASSP-35, pp. 400-401,*

Kay, M. 1987. "Non-concatenative Finite-State Morphology". Proceedings of EACL-87.

Kinoshita, S., J. Phillips, and J. Tsujii. 1992. "Interaction between Structural Changes in Machine Translation". *Proceedings in the 14th International Conference on Computational Linguistics*, vol 2, pp. 679-685, Nantes, France.

Kiraz, G. 1994. "Multi-tape Two-level Morphology". *Proceedings of COLING-94, 180–186.*

Koskenniemi, K. (1983). "Two-level morphology: a general computational model for word-form recognition and production". University of Helsinki, Department of General Linguistics, Publications, No. 11.

Kwon, H-C., and L. Karttunen. 1994. "Incremental Construction of a Lexical Transducer for Korean". Proceedings of COLING-94, 1262–1266.

Källgren, G. 1990. "The first million is hardest to get": Building a Large Tagged Corpus as Automatically as Possible. In: Karlgren, H. (ed.): *Papers presented to the 13th International Conference on Computational Linguistics*, vol. 3, pp. 168–173. University of Helsinki.

Lamel, L., M. Adda-Decker, and J.L. Gauvain. 1995. "Issues in Lage Vocabulary, Multilingual Speech Recognition," *Proceedings Eurospeech*, pp. 185-188.

Lee, C-H., B-H. Juang, and C-H. Lin. 1991. "A Study on Speaker Adaptation of the Parameters of Continuous Density Hidden Markov Models," *IEEE Trans. on Acoust., Speech and Signal Proc.*, Vol. ASSP-39(4), pp. 806–814.

Lee, C., R. Pieraccini, L. Rabiner, and J. Wilpon, "Acoustic Modeling for Large Vocabulary Speech Recognition," *Computer Speech and Language*, pp. 127-165.

Leggetter C.J., and P.C. Woodland. 1995. "Maximum Likelihood Linear Regression for Speaker Adaptation of Continuous Density Hidden Markov Models," *Computer Speech and Language*, pp. 171–185.

Lewin, I., S. Browning, D.M. Carter, K. Ponting, S.G. Pulman, and M. Russell. 1993. "A speech-based route enquiry system built from general-purpose components"

Life, A., and I. Salter. 1996. "Data Collection for the MASK Kiosk: WOZ vs Prototype System." *Proc. ICSLP '96*, Philadelphia.

Linell. P. 1981. "Speech errors and the Grammaticla Planning of Utterances". In Koch, W., C. Platzack, and G. Tottie (eds.): *Textstrategier i tal och skrift*, pp. 134–151. Almqvist & Wiksell, Stockholm.

Linguistic Data Consortium. http://www.ldc.upenn.edu/

McCord, M. 1993. "Heuristics for Broad-Coverage Natural Language Parsing." Proceedings of 1st ARPA Workshop on Human Language Technology, Princeton, NJ. Morgan Kaufmann.

MacDermid, C., M. Goldstein. 1996. "The 'Storyboard' Method: Establishing an unbiased vocabulary for keyword and voice command applications". *Proc. HCI'06*, London, August 1996. (In press.)

Mellish, C.S. 1988. "Implementing Systemic Classification by Unification". *Computational Linguistics* 14:40–51.

Meyer, E.A. 1937. "Die Intonation im Schwedischen, I: Die Sveamundarten". *Studies Scand. Philol. Nr 10*. University of Stockholm.

Meyer, E.A. 1954. "Die Intonation im Schwedischen, II: Die norrländischen Mundarten." *Studies Scand. Philol. Nr 11*. University of Stockholm.

Miller, S., R. Bobrow, R. Schwartz, and D. Stallard. 1996 "A Fully Statistical Approach to Natural Language Interfaces". *Proceedings of ACL-1996*, 55-61.

Miller, P. and I. Sag. 1995. "French Clitic Movement Without Clitics or Movement." CSLI Technical Report.

Mitchell, T., S. Kedar-Cabelli and R. Keller. 1986. "Explanation-Based Generalization: a Unifying View." Machine Learning 1:1, pp. 47-80.

Murveit, H., J. Butzberger, V. Digalakis and M. Weintraub. 1993. "Large Vocabulary Dictation using SRI's DECIPHER(TM) Speech Recognition System: Progressive Search Techniques". In *Proceedings of ICASSP-93*.

Murveit, H., J. Butzberger, V. Digalakis, and M. Weintraub. 1993. "Large Vocabulary Dictation using SRI's DECIPHER(TM) Speech Recognition System: Progressive Search Techniques." Proceedings of Inter. Conf. on Acoust., Speech and Signal, Minneapolis, Mn.

Neumeyer, L., V. Digalakis and A. Sankar. 1995. "A Comparative Study of Speaker Adaptation Techniques", Proceedings of European Conference on Speech Communication and Technology, pp. 1127–1130, Madrid, Spain.

Oflazer, K. 1993. "Two-level Description of Turkish Morphology". *Proceedings of European ACL-93*.

Oflazer, K. 1994. *Spelling Correction in Agglutinative Languages*. Article 9410004 in cmp-lg@xxx.lanl.gov archive.

Paul, D.B. 1989. "The Lincoln Robust Continuous Speech Recognizer," *Proceedings ICASSP*, pp. 449-452.

Paul, D.B. 1992. "An efficient stack decoder algorithm for continuous speech recognition with a stochastic language model." *Proceedings of 1992 International Conference on Acoustics, Speech, and Signal Processing*, volume 1, pages 25–28, San Francisco.

Paul, D.B., and J. Baker. 1992. “The Design for the Wall Street Journal-based CSR corpus,” *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 357–362.

Pulman S.G. 1992. “Passives”. In *Proceedings of the 3rd Conference of the European Chapter of the Association for Computational Linguistics*, pp. 306-313, University of Copenhagen, Copenhagen, Denmark.

Pulman, S.G. 1992. “Unification-Based Syntactic Analysis.” In Alshawi (ed), 1992.

Rayner, M. 1988. “Applying Explanation-Based Generalization to Natural-Language Processing.” Proceedings of the International Conference on Fifth Generation Computer Systems, Kyoto, pp. 1267-1274.

Rayner, M. 1993. “Abductive Equivalential Translation and its Application to Natural Language Database Interfacing.” PhD thesis, Royal Institute of Technology/Stockholm University. Also SRI Technical Report CRC-052

Rayner, M. 1994. “Overview of English Linguistic Coverage.” In Agnäs *et al*, (1994).

Rayner, M. 1994. “English linguistic coverage.” In Agnäs *et al*, (1994).

Rayner, M., H. Alshawi, I. Bretan, D.M. Carter, V. Digalakis, B. Gambäck, J. Kaja, J. Karlgren, B. Lyberg, P. Price, S.G. Pulman, and C. Samuelsson. 1993. “A Speech to Speech Translation System Built From Standard Components.” Proceedings of 1st ARPA workshop on Human Language Technology, Princeton, NJ. Morgan Kaufmann. Also SRI Technical Report CRC-031.

Rayner, M., P. Bouillon, and D.M. Carter. 1995. “Using Corpora to Develop Limited-Domain Speech Translation Systems”. In *Proceedings of Translating and the Computer 17*, ASLIB, London.*

Rayner, M., P. Bouillon and D.M. Carter. 1996. “Adapting the Core Language Engine to French and Spanish”. In *Proceedings of NLP-IA*, Moncton, New Brunswick.*

Rayner, M., D.M. Carter, V. Digalakis and P. Price (1994). “Combining Knowledge Sources to Reorder N-Best Speech Hypothesis Lists”. In *Proceedings of the ARPA workshop on Human Language Technology*, Princeton, NJ.*

Rayner, M., and D.M. Carter. 1996. “Fast Parsing using Pruning and Grammar Specialization”. In *Proceedings of 34th Annual Meeting of the Association for Computational Linguistics*, pp. 223–230, Santa Cruz, CA.*

Rayner, M., E. Beshai, I. Bretan, S. Rydin, and M. Wiren. 1996. “Composition of transfer rules in a multi-lingual MT system”, Proceedings of workshop on “Future Issues for Multilingual Text Processing”, Cairns, Australia, August 1996.*

Rayner, M., I. Bretan, D.M. Carter, M. Collins, V. Digalakis, B. Gambäck, J. Kaja, J. Karlgren, B. Lyberg, P. Price, S.G. Pulman and C. Samuelsson, “Spoken Language Translation with Mid-90’s Technology: A Case Study,” *Proc. Eurospeech ’93*, Berlin, 1993.*

Rayner, M., P. Bouillon, and D.M. Carter. 1995. “Using Corpora to Develop Limited-Domain Speech Translation Systems”. In *Proceedings of Translating and the Computer 17*, ASLIB, London.*

Rayner, M., and P. Bouillon, D.M. Carter. 1996. “Adapting the Core Language Engine to French and Spanish.” Proceedings of NLP-IA, Moncton, New Brunswick. Also SRI Technical Report CRC-061.*

Rayner, M., and D.M. Carter. 1996. “Fast Parsing using Pruning and Grammar Specialization”, Proceedings of ACL-96, April 1996.*

Rayner, M., D.M. Carter, V. Digalakis and P. Price. 1994. “Combining Knowledge Sources to Reorder N-Best Speech Hypothesis Lists.” Proceedings of 2nd ARPA workshop on Human Language Technology, Princeton, NJ., pp. 217-221. Morgan Kaufmann. Also SRI Technical Report CRC-044.*

Rayner. M., and C. Samuelsson. 1990. “Using Explanation-Based Learning to Increase Performance in a Large NL Query System.” Proceedings of DARPA Speech and Natural Language Workshop, June 1990, pp. 251-256. Morgan Kaufmann.

Rayner, M., and C. Samuelsson. 1994. “Corpus-Based Grammar Specialization for Fast Analysis.” In Agnäs *et al*, (1994).

Rayner, M., Alshawi, H., Bretan, I., Carter, D.M., Digalakis, V., Gambäck, B., Kaja, J., Karlgren, J., Lyberg, B., Price, P., Pulman, S. and Samuelsson, C. 1993. “A Speech to Speech Translation System Built From Standard Components.” Proceedings of 1st ARPA workshop on Human Language Technology. Also SRI Technical Report CRC-031.

Rayner, M. and P. Bouillon. 1995. “Hybrid Transfer in an English-French Spoken Language Translator.” Proceedings of IA ’95, Montpellier, France. Also SRI Technical Report CRC-056.

Rayner, M., D.M. Carter. 1995. “The Spoken Language Translator Project”, Proceedings of the Language Engineering Convention, London, July 1995.*

Rayner, M., and D.M. Carter. 1996. “Hybrid language processing in the Spoken Language Translator”, Proceedings of ICASSP-97, Munich, Germany.*

Rayner, M., P. Bouillon, D.M. Carter. 1995. Using Corpora to Develop Limited-Domain Speech Translation Systems. *Proc. Translating and the Computer 17 (ASLIB)*, November 1995.

Rayner, M., D.M. Carter, V. Digalakis and P. Price. 1994. “Combining Knowledge Sources to Reorder N-Best Speech Hypothesis Lists.” Proceedings of 2nd ARPA workshop on Human Language Technology.*

Rayner, M., D.M. Carter, P. Price and B. Lyberg. 1994. “Estimating the Performance of Pipelined Spoken Language Translation Systems.” Proceedings of ICSLP ’94, Yokohama.*

Rayner, M., D.M. Carter, P. Price, B. Lyberg. 1994. “Hybrid Transfer in an English-French Spoken Language Translator”, Proceedings of IA ’95, Montpellier, France, June 1995.*

Ritchie, G., G.J. Russell, A.W. Black and S.G. Pulman. 1992. *Computational Morphology*. MIT Press.

Ruessink, H. 1989. *Two Level Formalisms*. Utrecht Working Papers in NLP, no. 5.

Samuelsson, C. 1994. "Notes on LR Parser Design." Proceedings of COLING-94, Kyoto, pp. 386-390.

Samuelsson, C. 1994. "Grammar Specialization through Entropy Thresholds." Proceedings of ACL-94, Las Cruces, NM, pp. 188-195.

Samuelsson, C., and M. Rayner. 1991. "Quantitative Evaluation of Explanation-Based Learning as an Optimization Tool for a Large-Scale Natural Language System." Proceedings of 12th IJCAI, Sydney, pp. 609-615.

Sankar, A., and C-H. Lee. 1996. "A Maximum Likelihood Approach to Stochastic Matching for Robust Speech Recognition," *IEEE Transactions Speech and Audio Processing*, pp. 190–202.

Schwartz, R., and S. Austin. 1991. "A Comparison of Several Approximate Algorithms for Finding Multiple N-Best Sentence Hypotheses," Proceedings of ICASSP.

Shieber, S.M., G. van Noord, F.C.N. Pereira, and R.C. Moore. 1990. "Semantic-Head-Driven Generation." Computational Linguistics, 16:30–43.

Spies, M. 1994. "A Language Model for Compound Words in Speech Recognition". In *Proceedings of Eurospeech '95*, pages 1767–1770.

Tannen, D. 1982. "Oral and literate strategies in spoken and written narratives". *Language*, vol. 58. No. 1, pp. 1–21.

TMI. 1992. Proceedings of Fourth International Conference on Theoretical and Methodological Issues in Machien Translation. Montreal, Canada.

Tomita, M. 1986. Efficient Parsing for Natural Language. Kluwer Academic Publisher.

Trost, H. 1990. "The Application of Two-level Morphology to Non-Concatenative German Morphology". Proceedings of COLING-90, 371–376.

Trost, H. 1991. "X2MORF: A Morphological Component Based on Augmented Two-level Morphology". Proceedings of IJCAI-91, 1024–1030.

Trost, H., and J. Matiasek. 1994. "Morphology with a Null-Interface", Proceedings of COLING-94.

Woods, W. 1985. "Language Processing for Speech Understanding." Computer Speech Processing, W. Woods and F. Fallside (eds), Prentice-Hall International.

Yarowsky, D. 1994. "Decision Lists for Lexical Ambiguity Resolution". *Proceedings of 32nd Annual Meeting of the Association for Computational Linguistics*, pp. 88-95, Las Cruces, New Mexico.