Customizing Interaction for Natural Language Interfaces

Lars Ahrenberg, Arne Jönsson, Åke Thurée
Department of Computer and Information Science
Linkping University
S- 58183 LINKÖPING, Sweden
email: lah@ida.liu.se arj@ida.liu.se aketh@ida.liu.se

Abstract

Habitability and robustness have been noted as important qualities of natural-language interfaces. In this paper we discuss how these requirements can be met, in particular as regards the system's ability to support a coherent and smooth dialogue. The discussion is based on current work on customizing a dialogue system for three different applications.

We adopt a sublanguage approach to the problem and propose a method for customization combining bottom-up use of empirical data with a global pragmatic analysis of a given application. Finally, we suggest some design principles that have emerged from our work.

1 Introduction

Research on computational models of discourse can be motivated from two different standpoints. One approach is to develop general models and theories that apply to all kinds of agents and situations. The other is to develop accounts of specific discourse genres (Dahlbäck & Jönsson, 1992). It is not obvious that the two approaches should produce similar computational theories of discourse and we believe it is important to distinguish the two tasks from each other. Moreover, in the case of dialogues for natural-language interfaces (NLIs), which is our prime concern in this paper, there is not merely the question of modelling some external linguistic reality but also an important element of design, linguistic as well as otherwise.

The following requirements are widely recognized as being important for NLIs.

- habitability: the user should conveniently be able to express the commands and requests that the background system can deal with, without transgressing the linguistic capabilities of the interface (Watt, 1968);
- efficiency: the NLI should not slow down the interaction with the background system noticeably;

- robustness: the system should be able to react sensibly to all input (cf. Hayes & Reddy, 1983);

- transparency: the system's capabilities and limitations should be evident to the user from experience;

In this paper we discuss ways of making teletype natural-language interfaces satisfy these requirements in the context of applications which belong to the domain that Hayes and Reddy (1983) call simple service systems, i.e. systems that "require in essence only that the customer or client identify certain entities to the person providing the service; these entities are parameters of the service, and once they are identified the service can be provided" (*ibid.* p. 252). These systems exhibit the kind of dialogue that Van Loo and Bego (1993) term parameter dialogue.

The requirement that puts the highest demand on linguistic competence is the one concerning habitability. However, habitability does not necessarily imply that the system must understand any relevant request. Where extended linguistic coverage comes in conflict with either robustness, transparency or efficiency, it may be compromised (though cf. Ogden, 1988). This trade-off between requirements reconfirms the need for good design. The importance of habitability, however, suggests that a dialogue system must handle those phenomena that occur frequently in typed human-computer interaction correctly and efficiently, so that the user does not feel constrained or restricted when using the interface. The trade-off implies that the interface should not waste effort on complex computations in order to handle irrelevant or rare phenomena. For instance, the system need not be able to handle features such as jokes or surprise, when these are not demanded by the purposes of the system.

On these grounds we have adopted a sublanguage approach (Grishman & Kittredge, 1986) to dialogue systems of this kind. All aspects of linguistic communication, including interaction patterns and the use of indexical language is assumed to depend on the application and domain. For this reason the interface system must be designed to facilitate customization to meet the needs of different applications. Moreover, we must find methods that allow us to determine what the needs of a given application are. Such a method, based on Wizard of Oz-simulations, will be outlined below.

The rest of this paper is organized as follows. The next section briefly describes our system and applications, and the ways in which the system can be customized. The following section presents ways of meeting the requirements listed above as they apply to dialogue behaviour. In the third and final section we discuss our solutions from a more general perspective and propose three design principles relating to the notion of sublanguage, the quantity of information, and asymmetries between users and systems.

2 The linlin model

The natural language interface LINLIN (Ahrenberg, Jönsson, & Dahlbäck, 1990; Jönsson, 1991, 1993a); is designed to facilitate customization to various applications. Dialogue in LINLIN is modelled using dialogue objects which represent speech acts and speech act sequences. The dialogue objects are structured in terms of parameters that represent their properties and relations. A dialogue manager, which is the major controlling module of the system, records instances of dialogue objects as nodes of a dialogue tree as the interaction proceeds. The dialogue tree constitutes the global context of the dialogue.

The dialogue objects are divided into three main classes on the basis of structural complexity. There is one class corresponding to the size of a dialogue and another corresponding to the size of a discourse segment (cf. Grosz & Sidner, 1986). An initiative-response (IR) structure is assumed (cf. adjacency-pairs Schegloff & Sacks, 1973) where an initiative opens a segment by introducing a new goal and the response closes the segment (Dahlbäck, 1991). The third class corresponds to the size of a single speech act, or dialogue move. Thus, a dialogue is structured in terms of discourse segments, and a discourse segment in terms of moves and embedded segments. Utterances are not analysed as dialogue objects, but as linguistic objects which function as vehicles of one or more moves.[1]

2.1 Customization

Dialogue objects have been determined for three different applications on the basis of a corpus of 30 dialogues collected in Wizard of Oz-experiments (cf. Dahlbäck, Jönsson, & Ahrenberg, 1993; Fraser & Gilbert, 1991). 10 dialogues were used for each application. In one application, BILDATA, the system is a database providing information on properties of second-hand cars. The other two applications are both concerned with the travel domain. In one of them, TRAVEL1, users can only gather information on charter trips to the Greek Archipelago, while in the other, TRAVEL2, they can also order such a charter trip.

For the purpose of customization, two kinds of information can be obtained from a corpus:

- First, it can be used as a source of phenomena which the designer of the natural language interface was not aware of from the beginning, e.g. in the TRAVEL1 information system some users unexpectedly tried to make orders in spite of the fact that the system does not support it.

- Second, it can be used to rule out phenomena that do not occur in the corpus, especially those requiring sophisticated reasoning to be handled correctly.

[1] The use of three levels for the hierarchical structuring of the dialogue is motivated from the analysis of the corpora. There is no claim that they are sufficient for all types of dialogue, and even less so, to any type of discourse. The dialogue manager can be customized to deal with any number of levels.

Another matter is whether the corpus should serve as the only resource for determining what to include in the system, or whether the corpus data needs to be augmented somehow. The first, rather extreme stand is taken by Kelley (1983) who proposes a method, the User-Derived Interface (UDI), for acquiring the lexical and grammatical knowledge of a natural language interface in six steps. The first two steps are mainly concerned with determining and implementing essential features of the application. In the third step, known as the first Wizard of Oz-step, the subject interacts with what they believe is a natural language interface but which in fact is a human simulating such an interface. This provides data that are used to build a first version of the interface (step four). Kelley starts without grammar or lexicon. The rules and lexical entries are those used by the users during the simulation. In step five, Kelley improves his interface by conducting new Wizard of Oz simulations, this time with the interface running. However, when the user/subject enters a query that the system cannot handle, the wizard takes over and produces an appropriate response. The advantage is that the user's interaction is not interrupted and a more realistic dialogue is thus obtained. This interaction is logged and in step six the system is updated to be able to handle the situations where the wizard responded.

LINLIN is customized to a specific application using a process inspired by the method of User-Derived Interfaces. However, there is a drawback to Kelley's method as a very large corpus is needed for coverage of the possible actions taken by a potential user (cf. Ogden, 1988).

Our approach is thus less extreme. If a phenomenon is present in the corpus then it should be included. If it is not present, but occurs in other studies using similar background systems and scenarios and implementation is straightforward, the system should be customized to deal with it. Otherwise, if it is not present and it would increase the complexity of the system, it is not included. Knowledge from other sources is also employed (cf. Grishman, Hirshman, & Nhan, 1986). In the customization of LINLIN for the BILDATA and TRAVEL systems, knowledge on how the database is organised and also how users retrieve information from databases is needed.

3 Meeting requirements

In this section we discuss how the requirements listed in the introduction: habitability, efficiency, robustness and transparency, can be satisfied, in particular as they apply to the dialogue behaviour of the system.

3.1 Habitability

A dialogue object consists of parameters that specify different kinds of information. The parameters and their values can be modified for each new application, although some parameters are likely to be needed often. Customization of the Dialogue Manager to meet requirements on

habitability involves two major tasks:

- Defining focal parameters of the dialogue objects and customizing heuristic principles for changing the values of these parameters.
- Constructing a dialogue grammar for controlling the dialogue, i.e. specify parameters that determine what actions to take in different situations.

Two focal parameters, used in all three applications, are Objects and Properties. They are focal in the sense that they can be in focus over a sequence of segments. Their basic function is to represent the information structure of a move. Objects identify, via description or enumeration, a set of primary referents, and Properties identify a complex predicate ascribed to this set (cf. Ahrenberg, 1987).

Two principles for maintaining the focus structure are utilized. A general heuristic principle is that everything not changed in an utterance is copied from one IR-node in the dialogue tree to the newly created IR-node. Another principle is that the value for Objects will be updated with the value from the module accessing the database, if provided.

Primary parameters for defining the dialogue grammar are Type and Topic. Type represents the illocutionary force of a move. Hayes and Reddy (1983, p 266) identify two sub-goals in simple service systems: 1) "specify a parameter to the system" and 2) "obtain the specification of a parameter". Initiatives are categorized accordingly as being of two different types 1) update, U, where users provide information to the system and 2) question, Q, where users obtain information from the system. Responses are categorized as answer, A, for database answers from the system or answers to clarification requests. Other Type categories are Greeting, Farewell and Discourse Continuation (DC) (Dahlbäck, 1991). The latter type is used for system utterances that signal to the user that it is her turn.

Topic describes which knowledge source to consult. In our database applications three different topics are used: the background system for solving a task (T), the database model for queries about system properties, (S) and, finally, the dialogue tree for clarifications relating to the interpretation of moves (D).

A number of other parameters describing speaker, hearer, utterance content and so on, are also used. Although they provide additional information for the dialogue manager, the structure of the dialogue is largely captured through the parameters Type and Topic.

3.1.1 The focus structure

In the BILDATA application, task-related questions are about cars. Thus, the Objects parameter holds descriptions of, or explicit lists of cars while the Properties parameter, holds a set of car properties. In the TRAVEL systems, on the other hand, users switch their attention between

Statistics on focusing heuristics			
	BILDATA	TRAVEL1	TRAVEL2
Fully specified initiatives	52%	48%	47%
Initiatives requiring local context	40%	42%	45%
Initiatives requiring global context	4%	2%	5%

Table 1: User-initiatives classified according to context-dependence.

objects of different kinds: hotels, resorts and trips. This requires a more complex behaviour of the Objects parameter and the use of domain knowledge for focus tracking (Jönsson, 1993b).

The general focusing principles need to be slightly modified to apply to the BILDATA and TRAVEL applications. For the BILDATA application the heuristic principles apply well to the Objects parameter. An intensionally specified object description provided in a user initiative will be replaced by the extensional specification provided by the module accessing the database. For the TRAVEL applications the principles for providing information to the Objects parameter are modified to allow hotels to be added if the resort remains the same.

For the BILDATA application the heuristic principles for the Properties parameter need to be modified. The modification is that if the user does not add new cars to Objects, then the attributes provided in the new user initiative are added to the old set of attributes. This is based on the observation that users often start with a rather large set of cars and compare them by gradually adding restrictions (cf. Kaplan, 1983), for instance using utterances like *Remove all small-sized cars.* We call such responses *cumulative* as they summarize all information obtained in a sequence of questions. For the TRAVEL applications the copy principle holds without exception.

The results from the customizations showed that the heuristic principles worked well, minimizing the need to search the global context for referents of indexical constructions. See Table 1.

In the TRAVEL2 system there is one more object; the order form. A holiday trip is not fully defined by specifying a hotel at a resort. It also requires information concerning the actual trip: length of stay, departure date and so on. The order form is filled with user information during a system controlled phase of the dialogue.

3.1.2 The dialogue structure

The dialogue structure parameters Type and Topic also require customization. In the BILDATA system the users never update the database with new information, but in the TRAVEL2 system where ordering is allowed the users update the order form. Here another Type is needed, CONF,

Statistics on dialogue structure			
	BILDATA	TRAVEL1	TRAVEL2
Number of rules	15	12	14
Q_T/A_T	60%	83%	70%
Q_D/A_D	12%	2%	2%
Q_S/A_S	9%	2%	2%
Q_T/A_D	7%	2%	3%
Q_T/A_S	5%	6%	4%
Ordering rules	-	1%	17%
Others, e.g. greetings, farewells	7%	4%	2%

Table 2: Types of dialogue segments and their relative frequency in three different applications.

which is used to close an ordering session by summarizing the order and implicitly prompt for confirmation. For the ordering phase the Topic parameter O for order is added, which means that the utterance affects the order form.

The resulting grammars from the customizations of all systems are quite simple. The most common segment consists of a task-related initiative followed by an answer from the database, Q_T/A_T[2], sometimes with an embedded clarification sequence, Q_D/A_D. In BILDATA 60% of the initiatives start segments of this type. For TRAVEL 83% of the initiatives in the non-ordering dialogues and 70% of the ordering dialogues are of this type. Other task related initiatives result in a response providing system information, Q_T/A_S, or a response stating that the intitiative was too vague, Q_T/A_D. There are also a number of explicit calls for system information, Q_S/A_S. See Table 2 for a summary of the statistics on dialogue structure.

In the work by Kelley on lexical and grammatical acquisition, the customization process was saturated after a certain number of dialogues. The results presented here indicate that this is the case also for the dialogue structure of our applications. From a rather limited number of dialogues, a context free grammar can be constructed which, with a few generalizations, will cover the interaction patterns occurring in the actual application (Jönsson, 1993a).

The IR-sequences found in the analysis of dialogue structure have a natural explanation if we consider the purpose of the system. Although the dialogue objects do not represent information on user's goals, it turns out that the user utterances can be classified into a few classes in goal-related terms. The segments can basically be divided into four classes, taking the user's initiative as the basis for the classification: (i) "proper" information requests that are satisfied by an answer with information from the database, (ii) successful queries about system properties, (iii) successful moves satisfying subordinate goals, such as greetings or discourse continuations; (iv) initiatives that transgress the system's knowledge and which require robust error handling.

[2]For brevity, when presenting the dialogue grammar, the Topic of a move is indicated as a subscript to the Type. Labels of IR-segments have the form of a pair of move labels separated by a slash (/).

Note that we can view the taxonomy as a preference order. There is a basic division between the simple, normal segment (Q_T/A_T with two moves) and the other segments, which in one way or another indicates that we have left the normal smooth interaction, and do something subordinate to the main purpose. From the point of view of efficiency the interaction is optimal when it consists of a sequence of Q_T/A_T-segments. If the user resorts to a Q_S, there is something about the system that she isn't aware of, but which he could be aware of. The third type is also of a subordinate, instrumental character, but really unnecessary for an experienced user. Uninterpretable moves are obviously the least preferred ones.

3.2 Optimal responses in normal mode

If the system determines that a certain user initiative is a complete and successful Q_T, the only remaining task is to derive an appropriate SQL-query and respond with the tuples that are returned from the database. One problem that the system can encounter in this process is ambiguity: a certain word or expression may be translated to the query language in more than one way. For instance, in the BILDATA database there is information on acceleration and top speed, so it is not clear what a user has in mind if he asks whether a certain car make is fast (Sw. *snabb*), or requests to see a list of fast cars. Similarly, the adjective *rymlig* (Eng. spacious) may pertain to the booth or the space inside the car. In this case, one option is to enter into a clarification sub-dialog. However, this takes time and it may be more swift to provide information on all aspects that are potentially relevant. The user may easily ignore information, if he is not interested.

A similar strategy is used if a question is about the same set of cars as a previous question (cf. 3.1.1). Then the information requested in the second question is added to that of the first. Some subjects actually mentioned this as a good feature of the system in their comments, as it facilitates comparisons and evaluations to have all relevant information in a single table.

In both of these cases the system provides more information to the user than she has actually asked for. Some other cases where this happens are described in the following section.

3.3 Communication of meta-knowledge

As regards the second type of user initiative, the system queries, there is first the problem of distinguishing them from the task-oriented queries. Descriptively, these queries have a different topic. The topic is recognized from the major constituents of the query, in particular main verbs and their complements.

Frequently the user starts a session by asking a question such as *Vilka bilar finns? (What makes are there?)* or *Finns det priser p bde nya och begagnade bilar? (Can you provide prices on both new and second-hand cars?)* These questions are currently all answered by a pre-stored message giving a brief description of the contents of the database. This text usually gives more

information than the user has asked for, and, as with the second example above, provides an answer only implicitly. However, it would easily be possible to make the responses more fine-grained given a sufficient empirical basis. Another option would be to display this text in a separate window on the screen so that the question need not arise as part of the dialogue.

The most common type of system query in the corpus concerns how information should be interpreted, e.g. *Frklara siffrorna fr rostbengenhet. (Explain the numbers indicating susceptibility to rust)*. These are recognized on the basis of the occurrence of a predicate that indicates that the interpretation of a certain value, or type of value, is at stake. The answers are then not obtained from the BILDATA database, but from a special file forming a part of the database model. Thus, there is one text informing on how rust is evaluated and what the various numbers in the rust column means, which is used as a response to all questions of type S relating to rust. It should be noted that the strategy of answering system queries by pre-stored standard answers has the effect that the analysis of a system question need not be as detailed as the analysis of a task-oriented question.

3.4 Robustness

Although the model offers a general way to support robustness, i.e. that of engaging in a clarification sub-dialogue, this way is not always the best one. Many problems cannot be diagnosed correctly by the system, and other problems, such as mis-spellings, can be regarded as too small to warrant a clarification sub-dialog. Moreover, the model is restricted to information signalled verbally as part of a NL-dialog. But the system may communicate with the user by other means, e.g. in another window (pop-up window) or by non-verbal signals in the dialogue window. The purpose of this extra channel would be to give instructions and hints that enable the user to stay within the bounds of what the system allows with as little disturbance and time loss as possible. That is, it can be argued that it is preferable to solve problems by giving signals to the user that he need not respond to verbally, only interpret.

Below we discuss several types of transgressions that the user can make and discuss means to handle them. They relate to the lexicon, the sentence grammar, the dialogue grammar and the domain model, respectively.

It is common that the user enters a word that has no analysis in the lexicon. Considering the requirements on efficiency it is important that the transgression is detected as soon as possible. If lexical processing does not start until the user enters a carriage return, a lot of time may be wasted. The best one can achieve in this regard is to detect the mistake as soon as the user has pressed a character key that results in a substring with no continuation in the lexicon. As our current lexicon does not support error detection at this level, we go for the second best alternative, i.e. to detect errors when a word separator has been encountered. The error is presently signalled to the user by changing the font of the echoed input. There is no diagnosis

performed by the system; this is a task given over to the user on the assumption that he knows better than the system what could be the trouble.[3] We believe that a non-verbal signal is more adequate than a verbal message. The important thing is to make the user aware of the lexical transgression while he is still engaged in writing.

Grammatical transgressions are handled similarly to lexical transgressions paying due regard to early detection. How soon you can detect a grammatical error depends on the parsing technique you use. To increase efficiency we are using a left-to-right, incremental chart-parser, which means that parsing starts as soon as the user enters some input (Wirén, 1992; Wirén & Rönnquist, 1993). The time interval between the user's pressing the carriage return key and the moment when parsing is finished is thus reduced noticeably. For an input to receive a grammatical analysis it is necessary that every word of the input receives an analysis as part of some phrase, possibly as a direct descendant of the category spanning the whole input. Thus, if a word is entered that cannot combine with active edges to its left, nor yield a phrase that can combine with those edges (top-down filtering is used), then the system knows that no complete parse can be obtained. Thus, many grammatical transgressions can be detected fairly quickly. Currently there is no diagnosis but the transgression is signalled to the user as soon as it is detected by inverting the dialogue window. This shows the user that the input cannot be interpreted, although it does not tell him what is at fault. We are contemplating giving information of the following kinds: information about expressions that could follow strings that are left unextended at the relevant vertex, and lists of example sentences that are somehow similar to what the user has written, e.g. that contain the same initial words.

Dialogue grammar transgressions occur when the user enters something which the system can only interpret as a move which is out of place. For example, the user may interpret a system message as a question and enter "Yes", while the system intended it as a discourse continuation and expected a question. The transgression is detected when the dialogue manager tries to connect the user move with the current dialogue tree. It is signalled to the user by an explicit error message, which entails a superficial analysis of the error and brings recovery with it. The system takes no notice of the error, however, and does not enter the segment in the dialogue tree. This means that the segment is treated as if it had never occurred for the purpose of focus tracking and expectations, i.e. in the same fashion as a lexical or grammatical transgression. The dialogue continues in the same state as before the transgressive move.

Another type of transgression is when the user requests information that is not in the database, although the user seems to believe so. Some examples in the BILDATA domain concern the colour of cars, or variation in the number of doors. This type of error is detected by using information in the database model, which also should represent common knowledge of

[3]Of course, this need not always be the case and we are currently looking into robust parsing techniques to support diagnosis and recovery (Ingels, 1993). However, under current circumstances, the user is better equipped to handle lexical transgressions.

the domain e.g. properties of cars that people like to inquire about. These properties can be added to the database model as they are discovered and be marked there as not corresponding to anything in the database. Thus, these transgressions are easily detected and diagnosed. The response is an A_S-move and is taken to be part of the dialogue, as the user is likely to refer to objects and properties mentioned in his request afterwards.

A question may refer to a set of properties, or to several objects, only one of which is not in the database. The general strategy is then to filter out those properties and objects that there is no information on, but supply an answer for those that are correct. In addition to the retrieved table, an error message is included informing about the transgression. This is in line with the preference ordering of IR-segments: we prefer database information to be exchanged as much as possible.

4 A set of principles for design

In this section we ask whether the solutions to the specific problems that we have discussed, conform to some general design principles. Indeed, we believe that they do. Before looking at the principles we state some simple observations on the pragmatics of NL-dialogue with computers.

The importance of analysing the purpose of an application was noted in section 2.3. By considering the global discourse purpose we can get ideas on what kinds of discourse objects are likely to occur in a given application. This analysis can then be corroborated against empirical data. For our applications it seems that the goals of specific moves are either identical with the general discourse purpose, though constrained w r t the information requested, or subordinated to the general purpose. As we saw above it may happen, though, that users may attempt to pursue goals for which the system is only meant to be instrumental, as in the case of our subjects that tried to order tickets from an information system.

The topics that users may raise can be determined from an analysis of system requirements, which can then be tested empirically. As noted in section 3.1, we have found that topics in our data can be classified into three different classes, and all of them can be explained on a rational basis. As regards sub-classes of task-related topics, they depend very much on the domain.

An obvious, but nevertheless important observation on NLI:s and current NLP technology, is that the user and system have different abilities and knowledge. We note especially the following interesting asymmetries between the two participants of a NLI dialogue:

- The user reads and understands natural language at good speed; morever he is able to select information on the basis of relevance. In contrast, understanding is a hard task for the computer, not to talk about the task of distinguishing important from irrelevant information;

- The user is often slow at writing on the terminal. In contrast, the interface can display a lot of information on the screen in virtually no time at all.

- The system's knowledge of the standard language and its common sense knowledge of the world is only partial, while the user's knowledge is much higher. This would be true as far as language is concerned, even if we did not adhere to the sublanguage principle. Moreover, it is difficult to make the system adapt to users with different linguistic and knowledge skills, while users generally have this capacity to adapt to the competence of others. We can all adapt more or less to the language skills of children, foreigners, people with regional accents, and so on.

The general character of the interaction, the fact that one participant is a computer system and the other a human and that the human partner is involved in a specific (work) task are factors that all have important effects on the language used. The specific application and the domain have similar effects. Apart from the general conclusion that we have a more restricted development task to cope with than full NL-understanding and -interaction, we may also see it as a virtue to restrict the linguistic and conceptual knowledge of the system to that which is needed for the task at hand, since the interpretations that the system makes of the user's utterances can be fewer (less ambiguity) and more specific (domain-specific meanings and categorizations can be used).

We refer to this guide-line as **The Sublanguage Principle:** Restrict the linguistic and general knowledge of the system to that which is needed to support the users' tasks.

The Sublanguage Principle raises the question of what the necessary requirements are. In the end this problem requires an empirical answer, and the method of customization of dialogue structure and focus structure that we have described above is designed so as to find these answers. Our results so far indicate that for simple service systems the answers can be obtained with reasonable effort.

We also take The Sublanguage Principle to imply that the semantics of the language used is jointly provided by the background system and the dialogue objects. Thus, expressions are interpreted in terms of their contribution to a dialogue object of one of the recognized types. Meaning distinctions that are not relevant to the dialogue objects (and thus, by implication, to the system's behaviour, at large) are not recognized. Instead, expressions that have clearly different functions in the standard language may be treated as formal variants: Conventional expressions of indirectness, *Kan jag f (Can I have)*, *Jag skulle vilja (I would like)*, and so on, are treated as formal variants of the direct interrogative constructions. The system does not recognize politeness, i.e. it makes no difference between users w r t their formality or manners. The primary factor in determining what is to be included in the lexicon and grammar is frequency and interpretability. This means that we definitely accept constructions that may be regarded

as ungrammatical in the standard language, e.g. omitting prepositions in queries such as *Pris Mercedes? (Price Mercedes?)*, when they are frequent and easy to interpret.

Another case of superficial semantic analysis is given by our current analysis of the words *annan, annat, andra (meaning 'other')* as in the query *Vilka andra bilar under 70000 kr har samma eller bttre rostvrden? (What other cars below 70000 kr have the same or better figures for rust?)*. What we do in this case is simply ignoring the occurrence of the word in the semantic interpretation, though it is allowed as a formal option. The result is that the answer will show information also for the car(s) that served as the index for 'other'. This is not harmful, though, as the user is often interested in those cars for the sake of comparison.

This particular solution is also motivated by another design principle, which we term **The Quantity Principle:** The system may give more information to the user than has actually been requested provided it is potentially relevant. We have seen this observation applied in several other cases already, e.g. in responses to ambiguous or vague queries and in the use of cumulative answers (cf. 3.3 and 3.4). The principle itself can be motivated from what we know about the average user: she has the ability to select (within limits, of course) what is relevant for her. Moreover, for information in tabular form, selection does not require excessive reading, either.

The Quantity Principle generally increases speed and supports robustness, while there may be a conflict with transparency, if the principle is not applied with care. We also note that it contradicts Grice's (1975) second maxim of quantity: "Do not make your contribution more informative than is required". But then, Grice did not have human-computer dialogue in mind when he stated it.

We have noted several asymmetries between system and user. The guide-line that can be derived from those observations is simply that you should look for solutions that exploits the relative strengths of the two sides. More formally stated, we have **The Asymmetry Principle:** If there is a choice, prefer solutions that make the user learn from system contributions to solutions that require the system to learn from the user's contributions. This principle must be used with some care, of course, and we must recognize that technical developments may make NLI:s more powerful than they are today. For the present we generally want to minimize the inference tasks that the system has to cope with, such as diagnosing users' errors, inferring accurate representations of the user's goals and beliefs, instead relying on simpler means that reveal the capabilities and limitations of the system, e.g. by providing examples of what the system can understand or by displaying help messages.

5 Conclusion

The proposed method of customization and the design principles should be regarded as tentative as no real users have actually put their hands on our system. Yet, taken together, the pragmatic

observations on NLIs and the data from NLI dialogues that we have analysed suggest to us that many features, e.g. user models and general text inference components, that have been argued are important for good functionality of NLIs are not generally required for simple service applications. Instead our principles take us in another direction. Rather than modelling the user we prefer the system to present a good model of itself (The Asymmetry Principle) and rather than inferring interpretations we rely on application-specific meanings (The Sublanguage Principle). To put it in slogan-like phrase: if there is a choice, prefer global pragmatics at design time to local pragmatics at run-time.

Acknowledgements

This work results from a project on Dynamic Natural-Language Understanding supported by The Swedish Council of Research in the Humanities and Social Sciences (HSFR) and The Swedish National Board for Industrial and Technical Development (NUTEK) in their joint Research Program for Language Technology. We are indebted to Nils Dahlbäck and the other members of NLPLAB for many valuable discussions on the topics of this paper.

References

Ahrenberg, L. (1987). *Interrogative Structures of Swedish. Aspects of the Relation between grammar and speech acts.* Ph.D. thesis, Uppsala University.

Ahrenberg, L., Jönsson, A., & Dahlbäck, N. (1990). Discourse representation and discourse management for natural language interfaces. In *Proceedings of the Second Nordic Conference on Text Comprehension in Man and machine, Täby-Stockholm, Sweden, April 25-27, 1990.*

Dahlbäck, N. (1991). *Representations of Discourse, Cognitive and Computational Aspects.* Ph.D. thesis, Linköping University.

Dahlbäck, N. & Jönsson, A. (1992). An empirically based computationally tractable dialogue model. In *Proceedings of the Fourteenth Annual Meeting of The Cognitive Science Society, Bloomington, Indiana.*

Dahlbäck, N., Jönsson, A., & Ahrenberg, L. (1993). Wizard of oz studies - why and how. In *Proceedings from the 1993 International Workshop on Intelligent User Interfaces, Orlando, Florida.*

Fraser, N. & Gilbert, N. S. (1991). Simulating speech systems. *Computer Speech and Language*, *5*, 81–99.

Grice, P. H. (1975). Logic and conversation. In Cole, P. & Morgan, J. L. (Eds.), *Syntax and Semantics (vol. 3) Speech Acts*. Academic Press.

Grishman, R., Hirshman, L., & Nhan, N. T. (1986). Discovery procedures for sublanguage selectional patterns: initial experiments. *Computational Linguistics*, *12*(3), 205–215.

Grishman, R. & Kittredge, R. I. (1986). *Analysing language in restricted domains*. Lawrence Erlbaum.

Grosz, B. J. & Sidner, C. L. (1986). Attention, intention and the structure of discourse. *Computational Linguistics*, *12*(3), 175–204.

Hayes, P. J. & Reddy, D. R. (1983). Steps toward graceful interaction in spoken and written man-machine communication. *International Journal of Man-Machine Studies*, *19*, 231–284.

Ingels, P. (1993). Robust parsing with charts and relaxation. In *NODALIDA '93: Proceedings from 9:e Nordiska Datalingvistikdagarna, Stockholm, June 1993*.

Jönsson, A. (1991). A dialogue manager using initiative-response units and distributed control. In *Proceedings of the Fifth Conference of the European Chapter of the Association for Computational Linguistics, Berlin*.

Jönsson, A. (1993a). *Dialogue Management for Natural Language Interfaces – An Empirical Approach*. Ph.D. thesis, Linköping University.

Jönsson, A. (1993b). A method for development of dialogue managers for natural language interfaces. In *Proceedings of the Eleventh National Conference of Artificial Intelligence*, pp. 190–195.

Kaplan, S. J. (1983). Cooperative responses from a portable natural language database query system. In *Computational Aspects of Discourse*, pp. 167–208. MIT Press.

Kelley, J. F. (1983). *Natural Language and Computers: Six Empirical Steps for Writing an Easy-to-Use Computer Application*. Ph.D. thesis, The Johns Hopkins University.

Loo, W. V. & Bego, H. (1993). Agent tasks and dialogue management. In *Workshop on Pragmatics in Dialogue, The XIV:th Scandinavian Conference of Linguistics and the VIII:th Conference of Nordic and General Linguistics, Göteborg, Sweden, August 16-17, 1993*.

Ogden, W. C. (1988). Using natural language interfaces. In Helander, M. (Ed.), *Handbook of Human-Computer Interaction*. Elsevier Science Publishers B. V. (North Holland).

Schegloff, E. A. & Sacks, H. (1973). Opening up closings. *Semiotica*, *7*, 289–327.

Watt, W. C. (1968). Habitability. *American Documentation*, *July*, 338–351.

Wirén, M. (1992). *Studies in Incremental Natural Language Analysis.* Ph.D. thesis, Linköping University.

Wirén, M. & Rönnquist, R. (1993). Fully incremental chart-parsing. In *Third International Workshop on Parsing Technologies, Tilburg, The Netherlands and Durbuy, Belgium.*