Language engineering systems 2

- More on stochastic models
- Syntactic-semantic representation
  - Functional Dependency Grammar
  - (Typed) feature-based grammars
- Semantic modules
  - Semantic roles
  - word sense disambiguation
- Generation

Creating a language model

- Collect a corpus
- Divide into test and development corpus
- Create your n-gram model for suitable n
  - smoothing, back-off
- Test on development corpus
- Test on test corpus

Smoothing in a bigram model

<table>
<thead>
<tr>
<th></th>
<th>han</th>
<th>hon</th>
<th>var</th>
<th>är</th>
<th>denn</th>
<th>hår</th>
<th>där</th>
</tr>
</thead>
<tbody>
<tr>
<td>han</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>hon</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>var</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>är</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>denn</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>hår</td>
<td>0</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
<tr>
<td>där</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>X</td>
<td>0</td>
<td>X</td>
</tr>
</tbody>
</table>

Simplest variant: add 1 to every element in the matrix.
Better variant: Good-Turing (multiply by Nc+1/Nc)
Entropy and perplexity

- Measures on prediction difficulty
- Entropy: \( H(X) = - \sum_{x \in X} p(x) \log_2 p(x) \)
- Perplexity: \( 2^H \)

Analysis: Cascading processors

- Tokenisation
- Part-of-speech tagging of tokens
- Chunking of tokens
- Relating / joining chunks
- Solving coreference relations
- Building discourse structure
- ...

Implementing cascades

- Composing finite-state transducers
  - (or similar modules using e.g. Perl)
- Enforcing same format on all communicated data
  - E.g. TIPSTER-architecture implemented in GATE
  - Jmf. Chart-parsing
Functional Dependency Grammar Parsing

- Based on Constraint Grammar (SWECG)
  - Rules are used to SELECT or REMOVE interpretations
  - The rules are applied if conditions on the context are fulfilled, e.g., REMOVE (V) IF (-1C DET)

- Processing phases:
  - Morphological analysis (all possible readings collected from lexicon)
  - Morphological heuristics (words that are not in the lexicon)
  - Morphological disambiguation (remove faulty readings)
  - Morpho-syntactic mapping (verb, head, modifier)
  - Syntactic disambiguation (remove faulty readings)

- Ambiguities may remain, but are kept at the lexical level

Functional Dependency Grammar

- FDG make dependencies explicit in a tree structure
- Context tests in the rules use information about the heads and the depending words, and valency information, to create explicit dependencies
- Valency describes the number and types of modifiers a word can have
- Initially partial trees are constructed (for the verbs)
- Iterative application of the rules eliminates most faulty interpretations

Feature structures

- Simplest untyped (PATR-like)
  - Open-ended as long as there are no conflicts
  - Atomic feature values (unifies only if identical)
  - No help from development environment to maintain intended constraints

- Typed feature structures (LKB-like)
  - All types, incl. atomic types, organised in a hierarchy
  - Each type needs a specified set of features
  - Each feature defined on one type (and valid for all sub-types)
  - Intended constraints are coded and enforced by development environment
Type hierarchies

```
phrase   sign
word    string
unary-phrase  binary-phrase
  /       /
 HEAD-DTR  NONHEAD-DTR
```

Typed feature structure

```
sentence          HEAD-DTR
  head-comp-phr   HEAD-DTR
    verb          ORTH "söver"
      HEAD-DTR
        NONHEAD-DTR
          head-comp-phr
            HEAD-DTR
              prep
                ORTH "på"
              NONHEAD-DTR
                unary-phr
                  HEAD
                    noun
                      ORTH "bildskärmen"
```

Parsing unification-based grammars

- Chart-parser
  - A separate set of slides treat chart-parsing
  - + Chapter 10 of Jurafsky & Martin
- Feature structure unifier
  - + Chapter 11 in Jurafsky & Martin
Chartparsing

Rules
(Grammar and Lexicon)

Input
(String)

Control

Chart
(monotone data structures, represents partial results)

Agenda
(List of things to do)

Linguistic Knowledge Builder (LKB)

- Development environment
  - Open source (http://wiki.delph-in.net/moin/LKBTop)
  - Typed feature structures
  - Parsing and generation
  - Profiling
- Resources
  - Large grammars
    - LKB (English)
    - JACY (Japanese)
  - Matrix grammars

LKB Windows
Semantic roles

- Predicate-argument structure
- The surface structure of the argument of verbs are linked to roles in the semantic representation
- Alternation is the possible mappings between grammatical function and semantic roles
- Verb can have selectional restrictions that restrict the type of argument to a suitable type of concept

Semantic roles – Thematic roles

- From general (proto-agent and proto-patient) to very deep/specific (for each type of event/verb)
- Common thematic roles:
  - Agent
  - Patient
  - Object
  - Instrument
  - Location
  - Source
  - Goal

Semantic roles – SR lists

- A list of the type of roles the arguments corresponds to
- Verbs can be categorised based on similar semantics and similar SR lists
- Example:
  - Break verbs: bend, fold, shatter, crack
  - Hit verbs: slap, strike, bump, stroke
  - Break: Agent, Instrument, Object
  - Hit: Agent, Instrument, Location
Semantic roles – Direct mapping

- Rules for syntactic realisation of argument structure
- Example
  - Agent in a (non-passive) sentence is in most cases Subject NP
  - Object is in most cases Subject if the verb is intransitive, and Direct object if the verb is transitive

Word sense disambiguation

- Most words are ambiguous
- Meaning depends on the context
- Approaches
  - Selectional restrictions (in semantic analysis)
  - Stand alone
  - Supervised ML
  - Bootstrapping ML
  - Unsupervised ML
  - Dictionaries
  - Taggers

WSD – Selectional restrictions

- Use restrictions on thematic roles
  - Susanna runs
  - The machine runs
- Utilise type hierarchies
- Example:
  - I'm looking for a restaurant that serve vegetarian dishes
- Problems
  - Violations
  - Approximations
  - Metaphors
WSD – Machine learning

- Input
  - Target word
  - Context
- Processing
  - POS tagged
  - Context modification
  - Stemming or morphological analysis
- Feature vector
  - Collocation
  - Co-occurrence

WSD – Supervised ML

- Input: vector + correct category/label
- Naïve Bayes classifier
  - Maximise the probability of a sense given the input vector
    - Sense = \( \text{argmax} \ p(\text{sense} | \text{vector};\text{sense}) \)
- Decision list
  - A sequence of tests created due to accuracy
- Problems
  - Requires large amount of data
  - Requires key

WSD – Bootstrapping ML

- Small training set used as seeds
- Create an initial classifier through supervised ML
- Generate new data with the classifier
- Iterative development of classifiers with improving coverage and accuracy
WSD – Dictionaries

- Use definitions of senses to find overlaps of the word to be disambiguated and the words in the context
- Problem
  - Definitions are usually short

Natural language generation

- NLG is the process of creating a written or spoken “text” to achieve a specific communicative goal
- Based on knowledge representation
  - A sentence
  - A paragraph
  - A text
  - An utterance in a dialogue
- NLG is not just reversed parsing, has its own problems and solutions

NLG – Techniques

- Canned text – ready made messages
- Templates – messages with holes that take different values
- Complete generation
NLG – Templates

- Congruency
  - [a, the] NOUN-SING is COLOUR
  - [an] NOUN-SING is COLOUR
  - [the] NOUN-PL are COLOUR
  - an NOUN-N-SING is COLOUR-N-SING
  - an NOUN-T-SING is COLOUR-T-SING
  - NOUN-PL is COLOUR-A
  - NOUN-DEF is COLOUR-A

NLG - Choices

- Content – user expertise, context
- Structures (rhetorical) – order, relations
- Choice of word, referring expressions, syntax, lexicalisation, aggregation, passive/active, etc.
- Realisation – morphology, linear order, punctuation, etc.

NLG - Lexicalisation

- Lexeme
- Synonyms
- Referring expression
  - Mary’s car / her car / Mary’s new car
  - Expression to introduce and refer back
NLG - Aggregation

- Remove redundancy
- Coordinate
- Group similar information
- Ellipsis
- Embedded segment
  - Subordinate clause

NLG – Traditional reference architecture (Reiter and Dale, 2000)

- Goal
- Background system
- Discourse Planning
  - Discourse plan
  - Discourse model
- Sentence Planning
  - Sentence plan
  - User model
  - Aggregation/laxicalisation rules
- Surface Realiser
  - Algorithm for referring expressions
  - Lexicon
  - Grammar

NLG – RAGS architecture

- Defines high-level data types, data models, ... but no specific pipe-line
- Considers 7 low level generation tasks
  - Lexicalisation
  - Aggregation
  - Rhetorical structuring
  - Generating referring expressions
  - Ordering
  - Segmentation
  - Centering/salience tracking

Natural Language Engineering Vol. 10: 3rd pp. 227-260.