Language engineering systems 1

- Aims and structure of the course
- Aspects of language engineering systems
- Discrete vs stochastic approaches to language modelling
- Issues in evaluation

Purpose

The participants of the course should acquire theoretical and practical knowledge of methods and techniques used in natural language engineering. More precisely:
- the ability to analyse and evaluate language engineering systems and system components,
- the ability to design and implement/customise systems and system components for various tasks,
- theoretical and practical knowledge of techniques used in various applications.

Course organisation

- 2 lectures
- 4 seminars
  - Articles
  - Presentation and discussion of labs
- Design, implementation and evaluation
  - 3 lab assignments
  - 1 project
Terminology (Cunningham)

- "CL is a part of the science of language that uses computers as investigative tools"
- "NLP is a part of the science of computation whose subject matter is computer systems that process human language"
- "LE is the discipline or act of engineering software systems that perform tasks involving processing human language"

Language engineering systems

- "computer systems that process language for some task usually other than modelling language itself"
- Different perspectives
  - Systems that manage a specific task (L-system)
  - System that are a module/component of a larger system (L-subsystem)
  - Systems that perform a task in a workflow

Systems from different perspectives

1. L-system (or component)
2. L-subsystem
3. System in a workflow
Aspects of LE systems

- **Purpose**
  - Functions, input/output, users, …

- **Language modelling**
  - General language vs sub-language
  - Monolingual, multilingual, …
  - Text type, domain, …

- **Design and development**
  - Language resources and data generation
  - Handling variation, ambiguities and error

- **Evaluation**

Resources

- **Corpora**
  - Training vs test

- **Lexica and grammars**
  - Machine-readable dictionaries
  - Lexical databases

- **Taxonomies, Thesauri, Ontologies**

Language engineering - components

- **Tokenizers**
- **Stemmers**
- **Lemmatizers**
- **Morphological analysers**
- **Part-of-speech taggers**
- **Chunkers**
- **Parsers**
- **Word-sense disambiguators**
- **Named-entity recognizers**
- …
Development – Lingware

- Grammar engineering
- Time and resource consuming
- Hard to update and modify
- Experts might not be available or disagree
- Data-driven learning (“Training”)
  - Increased portability
  - Good coverage of examples
  - Data can be sparse or expensive to create

Types of language processing

- Recognition and modelling
- Analysis
  - Classifying
  - Parsing
  - Interpretation
- Synthesis
  - Generation (from fact representations to natural language)
  - Translation (from a text written in a different language)
  - Dialogue management (from user’s input)
- Disambiguation
- Data (lingware) generation

In LE linguistic objects are primarily modelled as strings

Any object in the hierarchy can be analysed as a string of objects of lower rank.
Analysis (associates strings with other stuff)

- **String ~ Class**
  - boy is N

- **String ~ Structure**
  - boys > boy + s

- **String ~ Description**
  - boys > [cat=N, number=PL, case=NOM, …]

- **String ~ Meaning**
  - boys > \{x; BOY(x)\}

**Generation** means going in the reverse direction.

---

Two major approaches to language modelling

- **Discrete models**
  - Given an alphabet, A, the language (string set) is some subset of $A^*$, that can be characterized by a set of rules.

- **Stochastic models**
  - Given an alphabet, A, the language is $A^*$, coupled with a probability distribution $p(s)$ for $s$ in $A^*$.

---

Related distinctions

- Linguistic vs data-driven modelling
- Deep vs shallow analysis
- Discrete vs stochastic models
Discrete models

- Lists of objects e.g., word lists
- Finite-state automata / regular expressions
- Formal grammars
  - Context-free grammar
  - Dependency grammar
  - Augmented grammars
  - Linguistic formalisms: HPSG, LFG, ...

Functional Dependency Grammar

Output from FDG ("Nova asleep on the screen"):

<table>
<thead>
<tr>
<th>Word</th>
<th>Stem</th>
<th>Func Dep</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nova</td>
<td>nova</td>
<td>subj:&gt;2</td>
<td>%NH N SG NOM</td>
</tr>
<tr>
<td>sover</td>
<td>sova</td>
<td>main:&gt;0</td>
<td>%MV V PRES</td>
</tr>
<tr>
<td>på</td>
<td>på</td>
<td>advl:&gt;2</td>
<td>%AH PREP</td>
</tr>
<tr>
<td>bildskärmen</td>
<td>bildskärmen</td>
<td>pcomp:&gt;3</td>
<td>%NH N SG NOM</td>
</tr>
</tbody>
</table>

Phrase structure tree

```
S
  /\  \
/   \ /
/     /
NP    VP
  /\  /\  \
/  |
PN  V  PP
     /\  /
    /  \\
   Nova sover på bildskärmen
```
Feature structure

Stochastic models

- N-gram language models
- Weighted automata
- Hidden Markov models
- Probabilistic grammars
- ...

Simple models

- A priori models
  - Choose the event with the highest probability
  - With knowledge of related event O
    - Choose $E^*$ = argmax $p(O|E)$
    - Examples:
      - N-gram language models
        - $O$ is previous word and $w^*$ = argmax $p(w'|w_{i-1})$
      - Trigram model probabilities
        - $O$ is two previous words and $w^*$ = argmax $p(w'|w_{i-2}w_{i-1})$
N-gram models

- **Data sparseness**
  - All n-grams of interest cannot be found in the training corpus.
- **Solutions**
  - "Smoothing" of probability mass onto all n-grams;
  - Rely on shorter n-grams when data is unavailable ("backoff")

The noisy channel model

- **Bayes’ rule**
  \[ P(E|O) = \frac{p(E)p(O|E)}{p(O)} \]
  \[ E^* = \arg\max P(E|O) = \arg\max p(E)p(O|E) \]
  \( p(E) \) is the a priori model
  (usually an n-gram model if \( E \) is a string)
  \( p(O|E) \) is the channel model

Bigram model as automaton

- An n-gram model is a statistical automaton (Markov model) where states are associated with symbol sequences of length n-1.
- Bigram models (character sequences):
Hidden Markov Models

- A HMM is a weighted automaton with a probability distribution over symbols in every state (rather than just a single symbol)
- The Viterbi algorithm
  - Given an observed symbol sequence, \( O \), determines the most likely state sequence to produce \( O \).

Evaluation

- Why (purpose, user)
- What (aspects to evaluate)
- How (methods)

Evaluation - Purpose

- Adequacy
  - To what extent does the system fit the task (and/or a larger system)?
- Progress
  - How far has the system been developed compared to
    - requirements
    - best possible performance
Evaluation - Extent

- "Black-box"
  - Looks only at the relation between input-output
- "Glass-box"
  - Looks at (some) system properties

Evaluation - Focus

- Task – How well does the system perform a task?
  - Correct (key/fact, gold standard)
  - Standards (benchmarking)
  - Minimum (base line)
- Function – How well does the system perform a service in an activity?
- Usability – How easy is it to use the system?
  - What knowledge and resources are needed?

Evaluation – Criteria and measures

- Criteria
  - Informal descriptions of desirable system properties
- Measures
  - Formal definitions that corresponds to criteria
  - Methods for measurements
  - Process that leads to values for measures
Evaluation – Phases

- Task description
  - What are the properties of the system?
  - What properties are interesting?
- Planning
  - Decide on criteria, measures, methods to fulfill requirements
- Realisation
  - Create test data; perform measurements, data analysis and conclusions

Evaluation – Realisation

- Separate test data from training data
- Decide on the correct answer
- Differentiate judge from developer
  - "Gold standards" – data is decided beforehand and calculations can be done automatically
- Analyse results
  - Type of errors, cause of errors

Evaluation – Measures

- Classification
  - Accuracy
  - Degree of ambiguity
- Retrieval
  - Precision
  - Recall
  - F-measure
- Application
Evaluation measures: Recall, Precision

<table>
<thead>
<tr>
<th>System/Source</th>
<th>Correct</th>
<th>Fault</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>p</td>
<td>fp</td>
</tr>
<tr>
<td></td>
<td>fn</td>
<td>n</td>
</tr>
</tbody>
</table>

- p = positive
- fp = false positive
- n = negative
- fn = false negative

Recall = \(\frac{p}{p + fn}\)
Precision = \(\frac{p}{p + fp}\)

Evaluation measure: F-measure

F-measure combines recall and precision

\[
F = \frac{2 \times P \times R}{P + R}, \text{ eller mer generellt} \\
F = \frac{P \times R}{\alpha P + (1- \alpha)R}
\]

Evaluation in applications

- There be many correct answers
- Translation, Question answering, generation ...
- Hard to evaluate a components contribution to the whole
- Components later on might enhance or reduce
Evaluation – Other aspects

- Subjective evaluation
  - Experts or users
  - Expensive and time consuming
- Economic evaluation
- Resources
- Dynamic aspects
  - Scalability
  - Portability