Language engineering systems

- Aims and structure of the course
- Different types of NL-systems
- Issues in design and development
- Issues in evaluation

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"

Purpose

The participants of the course should acquire in depth 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 algorithms and techniques used in various NLP-applications.

Language engineering – Problem types

- String transfer: speech-to-text, text-to-speech, ...
- Classification: language, subject, genre, part of speech, ...
- Analysis: phrase recognition, parsing, ...
- Generation
- Response: discourse/dialogue, information retrieval, ...

Course organisation

- Lectures (in English)
- Seminars (in Swedish)
  - Articles
  - Presentation and discussion of labs
- Practical work
  - 3 labs
  - 1 Project assignment

Language engineering – Modelling

- Symbolic (rule-based) models
  - Automata
    - finite automata, transducers
  - Formal rule systems
    - grammars
  - Logic
    - FOPL, feature structures
- Probabilistic models
  - n-gram language models, HMMs, ...
Language engineering - Resources

- Corpora
- Lexica and grammars
  - machine-readable dictionaries
  - lexical databases
- Taxonomies, Thesauri, Ontologies

Systems from different perspectives

- (D) system (or component) → input → output
- (S) subsystem
- (W) system in a workflow

Language engineering - Specific problems

- Tokenisation
- Tagging (categorisation)
- Word sense disambiguation
- Reference resolution
- Collocations
- Regular expressions
- Parsing

Tasks in design and development

- The language task(s)
  - what is available, what needs to be created?
  - systems, algorithms, resources, ...
- Other aspects of functionality
  - speed, memory requirements, parameters, ...
- The integration/communication task(s)
  - input formats, output formats, APIs, ...
- The user interface task

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

Design and development - Issues

- "Ideology"
  - explicit design,
  - iterative design,
  - implicit design
- Modelling language
- Method for design
- Method for implementation of design
Development process

- Inception
  - Requirements analysis
  - Requirements specification and plan
- Elaboration
  - Analysis of existing systems and components
  - Choice of techniques
  - Domain model and use cases

Development - Modelling languages

- Mathematical notation
- Natural language
- UML
  - The Unified Modelling Language (UML) is the industry standard language for specifying, visualising, constructing, and documenting the artifacts of a software system. It simplifies the complex process of software design, creating a 'blueprint' for construction.
- Javadoc
  - ... a tool for generating API documentation in HTML format from doc comments in source code

Development - Use cases

- "A typical interaction between a user and a computer system"
- Properties
  - Capture a user functionality
  - Small or large
  - Achieve a goal for the user
  - Divides functionality in parts which can be used as a basis for design and planning of implementation

Development (Johansson et al.)

Conceptual design

Theory

Hypothetical

Framework adaptation

Framework

Other modules/Architecture

Module

Development process

- Construction
  - Design, coding, testing
  - Integration with existing components
  - Documentation
- Transition
  - Beta testing
  - System

Development - Conceptual design

- "Paperwork"
- Design document
  - Modularisation
  - Knowledge representation
  - Interfaces
**Infrastructures**

3 types of infrastructures:
- Framework
- Architecture
- Development environments

**System architecture – Development environments**

- "an implementation of an architecture that includes some graphical tools for building and testing applications"
- Symbiotic relationship to architecture

**Framework**

- "a set of prefabricated software building blocks that programmers can use, extend, or customise for specific computing solutions"
- Facilitates reuse of code and system architecture
- Complexity vs Understandability
- OO class libraries

**Linguistic components**

Two types of components:
- Linguistic resources/data (lingware):
  - Corpora, lexica, grammars,…
- Processing algorithms
  - Tagger, linkers,…

**Architecture**

- "system structure - organization of the software, assignment of responsibilities to components…"
- Implicit or Explicit
- Reference architectures for families of application in a domain (e.g., TIPSTER, DARPA Communicator)
- Facilitates requirements analysis, design, implementation, evaluation

**Development – Lingware**

- Knowledge / Grammar engineer (Lingist)
  - Good results
  - Time and resource consuming
  - Hard to update and modify
  - Experts might not be available
- Training
  - Increased portability
  - Good coverage of examples
  - Data can be sparse or expensive to create
Lingware design
- Access
- Format
- Annotation
- Formalisms for representation
- Indexing and retrieval

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

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

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

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 measure: Accuracy

\[ P = \text{number of correctly classified objects} \]
\[ N = \text{total number of objects} \]
\[ \text{Accuracy} = \frac{P}{N} \times 100\% \]

Accuracy = 0.75 (or 75%)  

Evaluation – Realisation

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

Evaluation measure: Ambiguity

\[ T = \text{number of classifications} \]
\[ N = \text{total number of objects} \]
\[ \text{Ambiguity} = \frac{T}{N} \]

Ambiguity = 1.5  

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>Correct</td>
<td>p</td>
<td>fp</td>
</tr>
<tr>
<td>Fault</td>
<td>fn</td>
<td>n</td>
</tr>
</tbody>
</table>

p = positive  
fp = false positive
fn = false negative
Recall = \( p / (p + fn) \)
Precision = \( p / (p + fp) \)
Evaluation measure: F-measure

F-measure combines recall and precision

\[ F = \frac{2 \times P \times R}{P + R}, \text{ eller mer generelt:} \]

\[ F = \frac{P \times R}{\alpha P + (1 - \alpha) R} \]

---

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