Korpus-Abfrage: Werkzeuge und Sprachen

Motivation
Lizentiatsarbeit:
A Corpus Query Tool for Automatically Annotated Corpora

• Corpus Query Tool
• Theoretical Part about Corpus Query

Overview
• Corpus Query Tools:
  – SARA
  – TIGERSearch
• Theoretical Considerations:
  – Parameters of Corpus Query
  – Corpus Query Languages
• My Own Corpus Query Tool

“Languages” of Corpus Query
• Scripting languages (perl, tgrep, etc.):
  – Not very intuitive or easy to use
• Corpus Query Languages
  – Formal construct designed to retrieve data from corpora
  – Emphasis on linguistic information
    (trade-off between linguistic correctness and performance)
• SQL
  – For database queries only

Corpus Query Tools: SARA
• SARA:
  “SGML-Aware Retrieval Application”
• Query Tool for British National Corpus
  (BNC: 100 Million words, PoS-tagged)
• Makes use of Corpus Query Language
• Graphical interface (“Query Builder”) as well as Corpus Query Language CQL

SARA Query Possibilities 1
• Word query
  – (e.g. ‘colour’ retrieves ‘colour’, ‘coloured’, ‘colouring’, etc.)
• Phrase query
  – ‘home _ centre’ retrieves ‘home loan centre’ or ‘home improvement center’
• Pattern query
  – ‘colo?r’ retrieves all instances of ‘color’ and ‘colour’
SARA Query Possibilities 2

- PoS-query
  - "colour"=NN1 retrieves all instances of 'colour' as a noun
  - "colour"=VVI retrieves all instances of 'colour' as infinitive
- SGML-query
  - '<body>' retrieves all instances of the SGML-tag '<body>'

SARA Query Builder

Query Builder: visual interface to create complex queries

- Scope node (left)
  - e.g. search within the scope of a single SGML-element <body>
- Content node (right)
  - Find 'colour' in combination with PoS-tag 'VVB' or 'VVI'
    (BNC Tagset: VVI is infinitive of lexical verb, VVB is base form of lexical verb, except infinitive)

SARA Result Display

SARA CQL 1: Atomic Query

- Atomic query:
  - A word, punctuation mark, or delimited string (e.g. jam, ?, “Mrs.”)
  - A word-and-PoS pair (e.g. "CAN"=NN1)
  - A phrase (e.g. “not in your life”)
  - A pattern (e.g. color?)
  - An SGML query (e.g. <body>)
  - Wildcard character _ (e.g. home _ center)

SARA CQL 2: Unary Operators

- Unary operators:
  - Case: $ operator makes query case-sensitive
  - Header: @ operator makes query search within headers as well as bodies of texts
  - Not: ! Operator matches everything which is not a solution to the query
    (e.g. “cat dog” finds occurrences of ‘cat’ not preceded by ‘cat’)
SARA CQL 3: Binary Operators

• Binary operators:
  – Sequence: blanks between two queries (e.g. cat dog)
  – Disjunction: operator | matches cases which satisfy either query (e.g. cat | dog)
  – Join: * (order matters) and # (order does not matter) operator match cases which satisfy both queries (e.g. cat * dog)

SARA Conclusion

• Disadvantages:
  – no syntactic mark-up in BNC -> retrieval options less complex
  – no “delexicalized” search options for PoS
  – output functions restricted
• Advantages:
  – SGML search options
  – query builder
  – **BNCWeb** refines BNC query

SARA

• Literature:
  – SARA handbook
• Internet Resources:
  – SARA trial version for 30 days at [http://sara.natcorp.ox.ac.uk/](http://sara.natcorp.ox.ac.uk/)
  – Simple Search online at [http://sara.natcorp.ox.ac.uk/lookup.html](http://sara.natcorp.ox.ac.uk/lookup.html)

Corpus Query Tools: **TIGERSearch**

• Two-part system: **TIGERRegistry** and **TIGERSearch**
• **TIGERRegistry**:
  import and preprocessing of corpora
• **TIGERSearch**:
  querying, display and export of query results
  corpora:
    – NEGRA treebank (10’000 syntactically annotated sentences)
    – other corpora converted to TIGERXML-format

TIGERSearch Architecture

**TIGERRegistry**

**Source:** Lezius and König 2000a:114
TIGERSearch
Description/Query Language 1

• TIGER Description Language serves two purposes:
  – to encode the syntactic annotation of the corpus
  – to define queries
• TIGER Description Language Levels:
  – node level
  – node relation level
  – graph description level

TIGERSearch
Description/Query Language 2

• Node level:
  – nodes are feature-value pairs (e.g. word="Farbe", pos="NN")
  – combination of nodes with Boolean expressions (e.g. [word="Farbe" & pos="NN"])”)
• Node relation level:
  – nodes are combined by the following two relations:
  – direct precedence (horizontal dimension)
  – direct dominance (vertical dimension, operator >) (e.g. [cat="PP"] > [pos="APPRART"])

TIGERSearch
Description/Query Language 3

Graph description level:
(restricted) Boolean expressions combine node relations
(e.g. [cat="VP"] > [pos="APPRART"] & [cat="VP"] > [pos="VVPP"])\n
Corpus Query Languages: Overview

• Formal construct designed to retrieve data from corpora
• Corpus query language depends on project; many different versions available
• Conflict between traditional linguistic description languages (i.e. grammar formalisms) and efficiency

• Literature:
  – Smith, George. 2002. “A Brief Introduction to the TIGER Sample Corpus”
• Internet Resources:
  – TIGER Project http://www.ims.uni-stuttgart.de/projekte/TIGER
Corpus Query Languages: Elements

Corpus query languages consist of the following elements:
• Symbols for constituents;
• Symbols to describe the order of these constituents (horizontally as well as vertically);
• Boolean operators to combine (sequences of) constituents;
• Further options such as case-sensitivity, number, etc.

General Parameters of Corpus Query

• Research question:
  query for word, syntactic constituents, statistical information, etc.?
• User:
  beginner, intermittent user, experienced user?
• Corpus annotation:
  plain text, PoS-tagged, syntactically annotated, semantic tags?

Technical Considerations of Corpus Query

• Data storage:
  plain text, XML-encoded text, NEGRA Export Format, database, etc.
• Architecture:
  local program vs. client/server-architecture
• Interface:
  textual input vs. graphical interface
• Output:
  KWIC, PoS-tags, syntactic structures, graphical output, lemmas, etc.

My Own Corpus Query Tool

• User: beginner (can be extended to professional user)
• Architecture:
  – webbased query interface (PHP & HTML)
  – MySQL database on server at IFI
• Graphical query interface
• Corpus storage and retrieval from a database

Database Systems

• A database is a logically coherent collection of data with some inherent meaning
• A database is administered by a database management system (DBMS)
• Data in a database is modeled in a scheme which describes their meaning (meta-data)
• Relational Database Systems are based on “tables”
Advantages of Database Systems

- Centralized realization of all database functions (such as data definition, data organization, data integrity, access to specific data) allows consistent access to data
- Integration of all data avoids redundancy
- Data is independent of applications
- Database systems take measures to guarantee data integrity and control of multiple users
- "meta-data" informs about structure of data

Relational Database Schema (excerpt)

<table>
<thead>
<tr>
<th>Word</th>
<th>Tag</th>
<th>TagSimple</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
<td>Id</td>
<td>Id</td>
</tr>
<tr>
<td>Sentenceld</td>
<td>Txt</td>
<td>Description</td>
</tr>
<tr>
<td>TagId</td>
<td>TagSimpleId</td>
<td>Description</td>
</tr>
<tr>
<td>ParentId</td>
<td>EdgelabelId</td>
<td>SemNameId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>SemTypeId</td>
</tr>
<tr>
<td></td>
<td></td>
<td>LemmaId</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Id</th>
<th>Primary Key</th>
<th>Foreign Key</th>
</tr>
</thead>
</table>

Relational Database: MySQL Tables (excerpt)

<table>
<thead>
<tr>
<th>table word</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>table tag</th>
</tr>
</thead>
<tbody>
<tr>
<td>Id</td>
</tr>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
</tbody>
</table>

SQL

- SQL (Structured Query Language) is a relational data definition and manipulation language
- SQL query structure:
  SELECT <attribute list>
  FROM <table list>
  WHERE <condition>
- example query for word "vom"
  SELECT Txt FROM word WHERE Txt="vom"

Query Possibilities

- Query for words
  - single word
  - word followed by word in variable distance
- Query for PoS-tags
  - single PoS-tag
  - PoS-tag followed by PoS-tag in variable distance
- Query for syntactical constituents
- Query for lemma
- Corpus-Browsing

Query Interface
Result Display

- Simple Query:
  - KWIC
  - with PoS-tags
- Advanced/Lemma Query:
  - full annotation in verticalized sentence table (see left)

My Own Corpus Query Tool: Conclusion

- Disadvantages:
  - restricted versatility of query
- Advantages:
  - easy handling
  - different types of result display
- Performance with large corpora?

Literature

- Literature:
  - Plaehn, Oliver. 1998. "Datenbank-Dokumentation."
- Internet Resources:
  - http://www.ifi.unizh.ch/chmerz/CorpusQuery/start.html