

ISMLA Session 1 - Introduction and Setup

Björn Rudzewitz

Tübingen University

October 23, 2017

Plan

- 1 Seminar Structure
- 2 Introduction
- 3 Language Analysis - Basis for Analysis
 - Why analyze ?
 - Data Collection
 - Role of Corpora
 - Selecting Data
 - Data Modeling
 - Corpus Annotation
- 4 UIMA Annotations
- 5 Technical Setup
- 6 Discussion

Seminar Structure

- lecture Monday and Wednesday 2 - 4 pm in 1.13
- 1st part of semester: lectures and exercises (Studienleistung¹)
- 2nd part of semester: project work and term paper (Prüfungsleistung)
- 9 ECTS Core CL Hauptseminar

<https://moodle02.zdv.uni-tuebingen.de/course/view.php?id=1854>

¹fail maximally 1 for passing

Two overarching technological goals:

- ① Language Analysis
- ② Application Building

- language analysis:
monolingual → multilingual
- multilingual: NLP challenges combining multiple languages or in different languages
- analysis: tools allowing to build comparable representations

⇒ Goal: use technology and solve multilingual problems on an *industrial-strength* level

Why analyze ?

Why should we analyze language ?

Why do we need annotations frameworks (presented in this seminar) ?

Why analyze ?

Why should we analyze language ?

Why do we need annotations frameworks (presented in this seminar) ?

- meta data to enrich original text
- provide a standardized abstraction over data
- enable search and index over classes of data
- find hidden patterns in information
- prepare data for further purposes
- automatic annotation since manual annotation of larger data volumes very time-consuming (experts, competence vs. performance, ...)

procedure of language analysis²:

- assign classes to observations
- find patterns in classes

the steps are not necessarily independent

²in this seminar

Annotations

Terminology

- “observations”: digital, computer-readable language representations
- “classes”: operationalized as annotations
- “annotation”: a tuple of start and end index over observations and $n \in \mathbb{N}$ features
 - minimally: name of annotation (= class name)

Example: tokenization

- task: detect words in input stream of observations
- concretely: find word/token boundaries
- technically: add *annotations* with the feature `name=token` to an observation
- in the process of annotating tokens, the definition, annotation procedure, and applicability of the term “word” might change

Example:

- real-world example: library
- few instances of repeated books
- books with similar topics in similar region/shelf
- new books are classified according to previous data
- all books in one shelf with a shared prefix
(annotation about the topic of the book)

→ unstructured data gets structured by assigning annotations based on observations/abstractions in the data

Question intensifies for multilingual analysis:

- How to ensure comparability/compatibility of annotations ?
- in our example: how to index books in different languages ?
How to know how to index them ?
- Which part of the definition and assignment procedure is language-specific ?
- Which level of abstraction is concrete enough to be used by tools but provides necessary abstractions ?

One step back:

- Which annotations are there ?
- How are the observations to be annotated structured ?
- How can we know how the annotations were performed ?

→ a **corpus** contains this information

What is a (linguistic) corpus ?

- collection of annotated observations
 - raw, original data
 - meta data
 - global: annotation guidelines, license, creator, ...
 - local: annotations on individual parts of the observation

- testing (linguistic ?) theories
- searching relevant examples
- training/testing systems
- but:
 - corpus is never fully representative or balanced
 - absence/infrequency of construction not necessarily indicator of ingrammaticality, etc.
 - negative data: no data in search results doesn't mean it doesn't exist

important when preparing your own data or making claims about it

Selecting Data

to consider before collecting data:

- What domain/task is the data needed for ?
- From what sources can the data be accessed ? Is it legally and technically possible ?
- How to select data representative for this domain ?
- How can the data be selected in a balanced fashion ?

Data Modeling

- given a domain and data, decide on a model and use it for annotating/creating the corpus
- data **modeling**
- models
 - cover/include some aspects of objects
 - other aspects are omitted
 - can include alternative information: e.g. meta data not present in original data

important when writing you own analysis components

- for aspects included in model: what classes (system) to use ?
- How is the system organized ?
- attention: there often is more than one valid model, adaptations might be necessary based on actual data at hand

→ **Document your decisions !**

Corpus Annotation

- raw data collection completed
- next step: bring structure to unstructured raw data
- add *meta data* to original data
- *but*:
 - What meta data to assign ?
 - What system for assigning meta data ?
 - How to represent the meta data ?

based on Detmar Meurers' slides on Corpus Annotation Hauptseminar WS 14/15

Why annotate meta data ?

- systematic assignment of meta data enriches original data
- search index over abstractions (data classes, instead of compiling out all surface forms and context)
- allows to learn from the data/gain insights
- annotated data can be reused by others (who possibly couldn't perform the annotation)

annotations allow for a qualitative and quantitative analysis

further considerations:

- size of corpus: manual, computer-assisted, automatic annotation
- annotation accuracy of humans and tools

What do we mean with meta data ?

- data describing the primary, raw data (e.g. author, creation date, tags, ...)
- computational linguistics: often linguistic annotations (sentence/word boundaries, POS tags, parse trees, ...)
- but: often general meta data (created when, by whom, where, ...)

Leech's maxims:

- seven maxims for corpus annotation
- useful guidelines/starting point in practical corpus creation
- in practice: more decisions need to be made

Principles

- 1 Encode meta data such that removing it to recover raw data is possible.
- 2 Encode meta data such that itself can be recovered from the raw data. Annotation guidelines need to be available to end users.
- 3 decisions for boundary cases need to be made clear (adequacy/reliability)
- 4 Meta data should encode by whom and how meta data has been encoded.
- 5 End users should be informed the annotations are a useful tool, but could contain errors/mistakes.
- 6 Annotations should ideally be based on established principles and theory-neutral.
- 7 An annotation schemes shouldn't be introduced as the absolute standard.

additional criteria to Leech in practice:

- output/corpus should be digital
- machine readable, easily interpretable by machines
- well-formed (system implemented only according to defined rules)
- non-proprietary, non-binary format (better)
- archivable

How to store the information *in practice* ?

- structured text
- inline markup
- standoff annotation
- data base

- UIMA = Unstructured Information Management Architecture³
- industrial-strength framework to bring structure to unstructured information
- technically and conceptually: adding annotations to observations

³cf. Ferrucci and Lally [2004]

UIMA annotation

- standoff annotation data
- transparently documented because defined in explicit type system
- available to every component in a processing pipeline at every moment
- form dependent or independent layers of meta data combined in one shared representation for one document
- incrementally build upon each other

→ fulfill described considerations from a technical and conceptual point of view

UIMA Annotations

Building rich representations

```
My parents always tell me what to do. Last Saturday I bought a new skirt, but they told me to take it back to the hop. They said: "It is too short." Have you ever heard of a 16-year-old who has to ask her parents about a skirt ? Right now I'm trying to do as much as possible in a way that my parents can't find about it. I sometimes hide my make-up at a friend's place.]
```

Figure: A plain text document

UIMA Annotations

Building rich representations

My parents always tell me what to do. Last Saturday I bought a new skirt, but they told me to take it back to the shop. They said: "It is too short." Have you ever heard of a 16-year-old who has to ask her parents about a skirt? Right now I'm trying to do as much as possible in a way that my parents can't find out about it. I sometimes hide my make-up at a friend's place.

Annotation Types

<input checked="" type="checkbox"/> ADJ	<input checked="" type="checkbox"/> ADV	<input checked="" type="checkbox"/> Answer	<input checked="" type="checkbox"/> ART	<input checked="" type="checkbox"/> CARD
<input checked="" type="checkbox"/> CompAdv	<input checked="" type="checkbox"/> CONJ	<input checked="" type="checkbox"/> Dependency	<input checked="" type="checkbox"/> DocumentM...	<input checked="" type="checkbox"/> KeyValuePair
<input checked="" type="checkbox"/> Lemma	<input checked="" type="checkbox"/> Morphologic...	<input checked="" type="checkbox"/> NN	<input checked="" type="checkbox"/> NP	<input checked="" type="checkbox"/> O

SoFa:

Mode: ☒ Annotations ☐ Features

Click In Text to See Annotation Detail

- Annotations
 - Lemma
 - Token
 - Token ("Saturday")
 - begin = 43
 - end = 51
 - lemma = Lemma ("Saturday")
 - begin = 43
 - end = 51
 - value = Saturday
 - pos = NP ("Saturday")
 - begin = 43
 - end = 51
 - PosValue = NNP
 - morph = MorphologicalFeatures ("id = null")

Figure: Example UIMA annotations on the document

Technical Setup

starting point for next week:

- Linux/Ubuntu operating system
- Java language and compiler
- Eclipse development environment
- Maven (IDE plugin)
- UIMA (IDE plugin)

you can also use different specs, but then we can't provide support

Install the UIMA plugin:

- provides UIMA-related user interface components and functions
- *Help* → *Install New Software ...*, then enter
`https://www.apache.org/dist/uima/eclipse-update-site/`
- Check for successful installation via
Help → *Installation Details*

What does Maven provide ?

- Java source code dependency management
- project build cycles:
 - compiling, testing, building, deploying, ...
- creating project stubs (archetypes)

references: official page: <https://maven.apache.org/what-is-maven.html>
“Maven Essentials” by Siriwardena [2015]

Install maven:

- command line:
`sudo apt-get install maven`
then run `mvn -version` to verify
- Eclipse:
Help → *Eclipse market place* → *Find m2e*
- the m2e plugin adds new menus and functions to Eclipse

newer versions of Eclipse might come with Maven already set up

- archetype⁴: project template
- Sets up a Java project with a pre-defined maven structure and task/archetype-specific additions

⁴[https:](https://maven.apache.org/guides/introduction/introduction-to-archetypes.html)

[//maven.apache.org/guides/introduction/introduction-to-archetypes.html](https://maven.apache.org/guides/introduction/introduction-to-archetypes.html)

Creating and building a simple example project:

```
mvn archetype:generate  
-DgroupId="de.ws1718.ismla"  
-DartifactId="demo"  
-Dversion="0.0.1"  
-DinteractiveMode="false"
```

```
mvn clean install
```

- Eclipse plugin comes with GUI for archetype selection

Maven

Project Structure

- pom.xml
- src/main/java
- src/main/resources
- src/test/java
- target

- all dependencies (artifact id, group id, version number)
- repositories
- information about the project itself
- plugins
- build settings
- . . .

Maven

Build Cycles

- predefined steps for building a project
- later steps automatically run earlier steps that are required for them
- default build cycle stages:
 - validate
 - compile
 - test
 - verify
 - install
 - deploy
- often useful: `clean` build cycle
- Eclipse: *Run as* → *Maven build* ...

source: <http://maven.apache.org/guides/introduction/introduction-to-the-lifecycle.html>

Homework

due next Monday:

set up a system with the described specification

in case of questions or problems: ask in course forum on Moodle

Discussion & Questions

- syllabus
- questions regarding the contents
- other points you want to discuss

David Ferrucci and Adam Lally. Uima: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3-4):327–348, 2004.

Prabath Siriwardena. *Maven Essentials*. Packt Publishing Ltd, 2015.