Apache UIMA

PU Machine learning mit Java, Weka und UIMA

Nils Reiter,
nils.reiter@uni-koeln.de

January 12, 2021
Winter term 2020/21

Section 1

Looking Back

Exercise 8

Section 2

NLP Data Structures

NLP Data

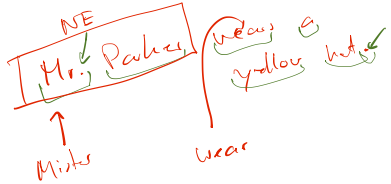
- Text and annotations
- Annotations: Meta data associated with specific locations in the text

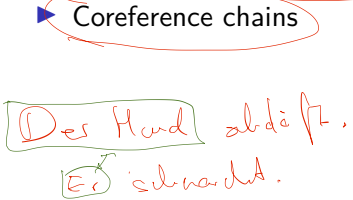
Storing Text

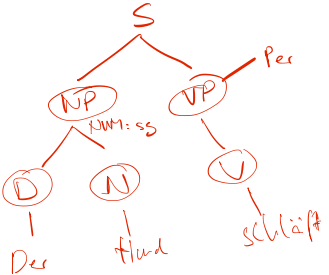
- Texts can be stored as strings
- Beware: Unicode
- (Theoretic) limitations on length

NLP Tasks

- Tokenization, sentence splitting
- Morphological analysis
- Part of speech-tagging, lemmatisation
- Named entity recognition
- Phrase structure and dependency syntax
- Coreference chains







NLP Tasks

- Tokenization, sentence splitting
- Morphological analysis
- Part of speech-tagging, lemmatisation
- Named entity recognition
- Phrase structure and dependency syntax
- Coreference chains

Tokens

- Many NLP data items rooted in tokens
  - E.g., named entities or phrases in syntactic annotation go over multiple tokens, coreference links are connect multiple multi-word expressions

Graph as Data Structure

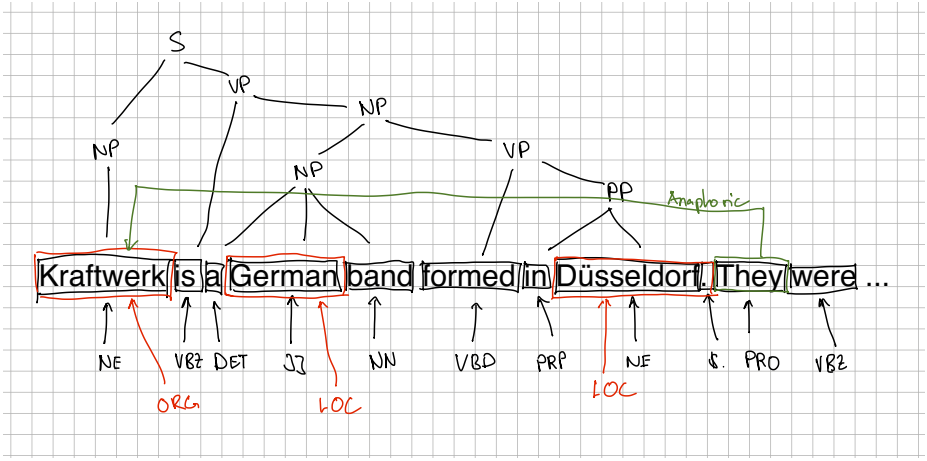


Figure: Beginning of Wikipedia article about Kraftwerk with (some) linguistic annotations

Section 3

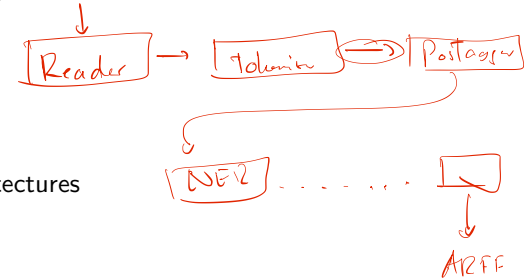
Apache UIMA

UIMA

- UIMA: Unstructured Information Management Architecture
- `https://uima.apache.org`
- Apache Project: Open source

UIMA

- UIMA: Unstructured Information Management Architecture
- `https://uima.apache.org`
- Apache Project: Open source
- Framework for
  - Data structures, with efficient access
  - Processing, in particular pipeline architectures
  - Modularization

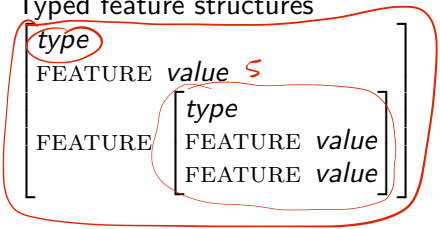


UIMA

- UIMA: Unstructured Information Management Architecture
- `https://uima.apache.org`
- Apache Project: Open source
- Framework for
  - Data structures, with efficient access
  - Processing, in particular pipeline architectures
  - Modularization
- Eco system
  - uimaFIT: Facilitates use of UIMA
  - DKPro: Prepackaged NLP components

UIMA Data Structures

- Typed feature structures



UIMA Data Structures

- Typed feature structures

$$\begin{bmatrix} \textit{type} & \\ \text{FEATURE} & \textit{value} \\ \text{FEATURE} & \begin{bmatrix} \textit{type} & \\ \text{FEATURE} & \textit{value} \\ \text{FEATURE} & \textit{value} \end{bmatrix} \end{bmatrix}$$

- Each FS represented as an object of class *type*
  - E.g., the class `Token` represents a FS of type 'Token'
  - Similar to java beans
- Types define which features instances of the type have
- Regular class hierarchy for inheritance

UIMA Data Structures

- Typed feature structures

$$\begin{bmatrix} \textit{type} & \\ \text{FEATURE} & \textit{value} \\ \text{FEATURE} & \begin{bmatrix} \textit{type} & \\ \text{FEATURE} & \textit{value} \\ \text{FEATURE} & \textit{value} \end{bmatrix} \end{bmatrix}$$

- Each FS represented as an object of class *type*
  - E.g., the class `Token` represents a FS of type 'Token'
  - Similar to java beans
- Types define which features instances of the type have
- Regular class hierarchy for inheritance
- Types are defined in XML files ⚠

UIMA Data Structures

Pre-defined types

- `uima.cas.TOP`: Super type for all types
  - Only feature: JCAS – the 'document'

UIMA Data Structures

Pre-defined types

- `uima.cas.TOP`: Super type for all types
  - Only feature: JCAS – the 'document'
- `uima.cas.Annotation`: Super type for all annotation types
  - Inherits from `uima.cas.TOP`
  - Additional features: BEGIN and END
    - Character positions w.r.t. the document
  - Additional method `getCoveredText()` to retrieve text surface

UIMA Data Structures

Defining types

- Type system description in XML file
- JCasGen: Generate Java classes from XML files (integrates nicely with maven!)

```xml
<?xml version="1.0" encoding="UTF-8"?>
<typeSystemDescription xmlns="http://uima.apache.org/resourceSpecifier">
  <name>NAME</name>
  <description />
  <version>1</version>
  <vendor>VENDOR</vendor>
  <types>
    <!-- ... -->
  </types>
</typeSystemDescription>
```

UIMA Data Structures

Defining types

```
1   <typeDescription>
2     <name>de.ukoeln.idh.teaching.jml.Token</name>
3     <description />
4     <supertypeName>uima.tcas.Annotation</supertypeName>
5     <features>
6       <featureDescription>
7         <name>PartOfSpeech</name>
8         <description />
9         <rangeTypeName>uima.cas.String</rangeTypeName>
10      </featureDescription>
11    </features>
12  </typeDescription>
```

UIMA Data Structures

Defining types

typeDescription

name	fully qualified Java class name
description	natural language description
supertypeName	the type we inherit from (`TOP`, `Annotation` or another defined by us)
features	features our type has

UIMA Data Structures

Defining types

typeDescription

name	fully qualified Java class name
description	natural language description
supertypeName	the type we inherit from (`TOP`, `Annotation` or another defined by us)
features	features our type has

featureDescription

name	feature name (will be used for `get-` and `set` methods)
description	natural language description
rangeTypeName	the type of our feature: `uima.cas.String`, `uima.cas.Integer`, `uima.cas.Double`, `uima.cas.Boolean` – or any other type

UIMA Data Structures

Collection types

- UIMA defines two types of collection types for feature structures
  - In both cases, `elementType` element can be used to describe type of collection elements

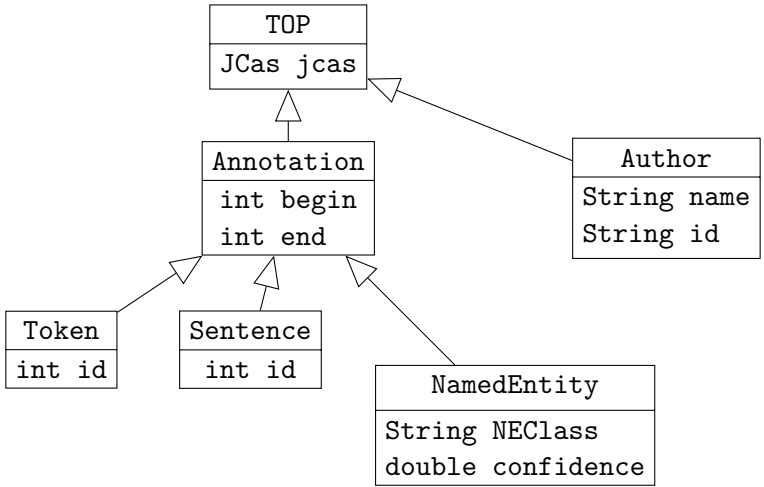
UIMA Data Structures

Collection types

- UIMA defines two types of collection types for feature structures
  - In both cases, `elementType` element can be used to describe type of collection elements
- `uima.cas.FSArray`, `uima.cas.StringArray`, `uima.cas.BooleanArray`, ...
  - Efficient access, length known at creation time
- `uima.cas.FSList`, `uima.cas.StringList`, `uima.cas.IntegerList`, ...
  - Linked lists: no random access, but variable length

UIMA Data Structures

Example

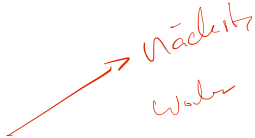


JCasGen: Generating Java Code from Type System Descriptions

- Documentation:
  `https://uima.apache.org/d/uimaj-current/tools.html#ugr.tools.jcasgen`
- GUI / command line program / Eclipse integration
- Maven plugin 👍
  - `mvn process-resources`
  - Allows to only commit XML files, not Java code

UIMA Data Structures

Access



- UIMA indexes all feature structures, such that we can efficiently access them
- This improved a lot with UIMA 3

```
1 SelectFSs<Token> selector = jcas.select(Token.class);
2 selector.asList(); // a list of all feature structures
3 selector.count(); // number of feature structures
4 selector.coveredBy(anotherAnnotation); // feature structures covered by another FS
5 selector.following(position); // first token after position
6 selector.get(index); // get nth token
7 selector.allMatch(predicate); // get tokens for which predicate is fulfilled
```

Section 4

Next Exercise

Exercise 09

`https://github.com/idh-cologne-machine-learning-mit-java/exercise-09`