ISMLA Session 6 - UIMA

Björn Rudzewitz

Tübingen University

November 27, 2017

Plan

1. Motivation
2. Terms
3. Components
4. Analysis Tasks
5. Type System

Motivation

- UIMA class exercises: everyone defined their own type system
- when using multiple tools: annotations build on each other (I/O capabilities)
- type system is not separable from analyis engines

Motivation

- UIMA class exercises: everyone defined their own type system
- when using multiple tools: annotations build on each other (I/O capabilities)
- type system is not separable from analyis engines

⇒ components are only exchangeable when also the type system is shared

Motivation

- problem of compatibility of types/annotations intensifies for
  - other tools
    - different input capabilities
    - different output format and granularity of features
  - other languages
    - different features needed for different languages

Motivation

DKPro provides a solution to this problem:

"Many NLP tools are already freely available in the NLP research community. DKPro Core provides Apache UIMA components wrapping these tools (and some original tools) so they can be used interchangeably in UIMA processing pipelines. DKPro Core builds heavily on uimaFIT which allows for rapid and easy development of NLP processing pipelines, for wrapping existing tools and for creating original UI components."[1]

[1] https://dkpro.github.io/dkpro-core/, last access 2017-11-27

Terminology

- tool
- component
- processing framework
- resource
- component collection

cf. [Eckart de Castilho and Gurevych, 2014]

Components

- DKPro Core[2] wraps a wide range of NLP tools as uimaFIT annotators
- all components use the same shared type system for interoperability
- components can be easily swapped because other components of the same category (e.g. tokenizer) produce the same types as output functions
- explicit and comprehensive type system provides most types for most tasks

[2] other DKPro projects exist, e.g. DKPro statistics

Components and Resources

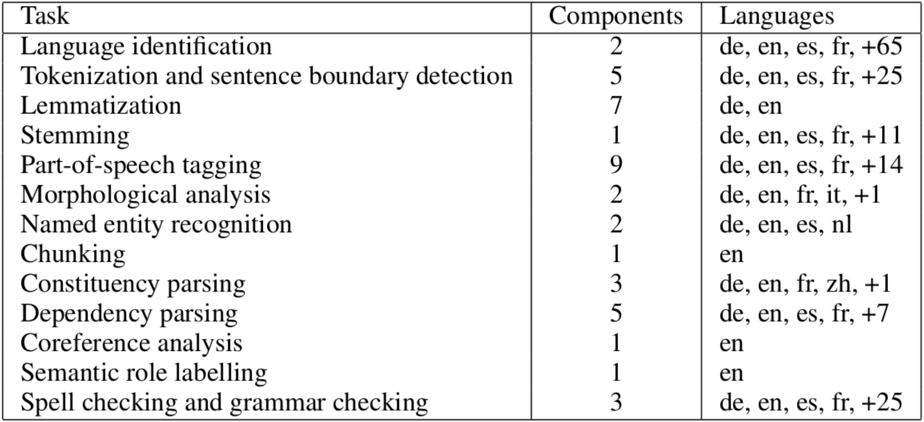
- all components and resources grouped in central Maven repository[3]
- custom components can be added building on the specification
- open-source implementation of components allows for easy lookup and extension of functions

[3] https://mvnrepository.com/artifact/de.tudarmstadt.ukp.dkpro.core

Resources

- resources (e.g. language-specific models) are described by coordinates:
  - tool
  - language
  - resource variant
  - version
- component coordinates are expressed as Maven coordinates

Analysis Tasks

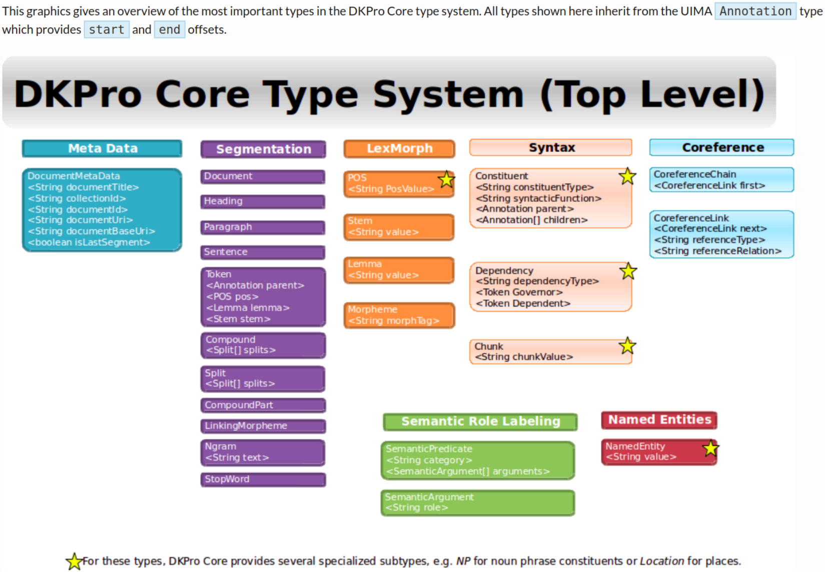


Task	Components	Languages
Language identification	2	de, en, es, fr, +65
Tokenization and sentence boundary detection	5	de, en, es, fr, +25
Lemmatization	7	de, en
Stemming	1	de, en, es, fr, +11
Part-of-speech tagging	9	de, en, es, fr, +14
Morphological analysis	2	de, en, fr, it, +1
Named entity recognition	2	de, en, es, nl
Chunking	1	en
Constituency parsing	3	de, en, fr, zh, +1
Dependency parsing	5	de, en, es, fr, +7
Coreference analysis	1	en
Semantic role labelling	1	en
Spell checking and grammar checking	3	de, en, es, fr, +25

Figure: DKPro components; table taken from [Eckart de Castilho and Gurevych, 2014, page 7]

Typesystem

https://dkpro.github.io/dkpro-core/releases/1.7.0/typesystem/



This graphics gives an overview of the most important types in the DKPro Core type system. All types shown here inherit from the UIMA `Annotation` type which provides `start` and `end` offsets.

DKPro Core Type System (Top Level)

Meta Data

DocumentMetaData
<String documentTitle>
<String collectionId>
<String documentId>
<String documentUri>
<String documentBaseUri>
<boolean isLastSegment>

Segmentation

Document

Heading

Paragraph

Sentence

Token
<Annotation parent>
<POS pos>
<Lemma lemma>
<Stem stem>

Compound
<Split[] splits>

Split
<Split[] splits>

CompoundPart

LinkingMorpheme

Ngram
<String text>

StopWord

LexMorph

POS ☆
<String PosValue>

Stem
<String value>

Lemma
<String value>

Morpheme
<String morphTag>

Syntax

Constituent ☆
<String constituentType>
<String syntacticFunction>
<Annotation parent>
<Annotation[] children>

Dependency ☆
<String dependencyType>
<Token Governor>
<Token Dependent>

Chunk ☆
<String chunkValue>

Coreference

CoreferenceChain
<CoreferenceLink first>

CoreferenceLink
<CoreferenceLink next>
<String referenceType>
<String referenceRelation>

Semantic Role Labeling

SemanticPredicate
<String category>
<SemanticArgument[] arguments>

SemanticArgument
<String role>

Named Entities

NamedEntity ☆
<String value>

☆For these types, DKPro Core provides several specialized subtypes, e.g. *NP* for noun phrase constituents or *Location* for places.

Exercise

see handout

DKPro References

DKPro Core [Eckart de Castilho and Gurevych, 2014]
web page: https://dkpro.github.io/dkpro-core/

information on the slides is partly based on these resources

Richard Eckart de Castilho and Iryna Gurevych. A broad-coverage collection of portable nlp components for building shareable analysis pipelines. In *Proceedings of the Workshop on Open Infrastructures and Analysis Frameworks for HLT*, pages 1–11, Dublin, Ireland, August 2014. Association for Computational Linguistics and Dublin City University. URL `http://www.aclweb.org/anthology/W14-5201`.