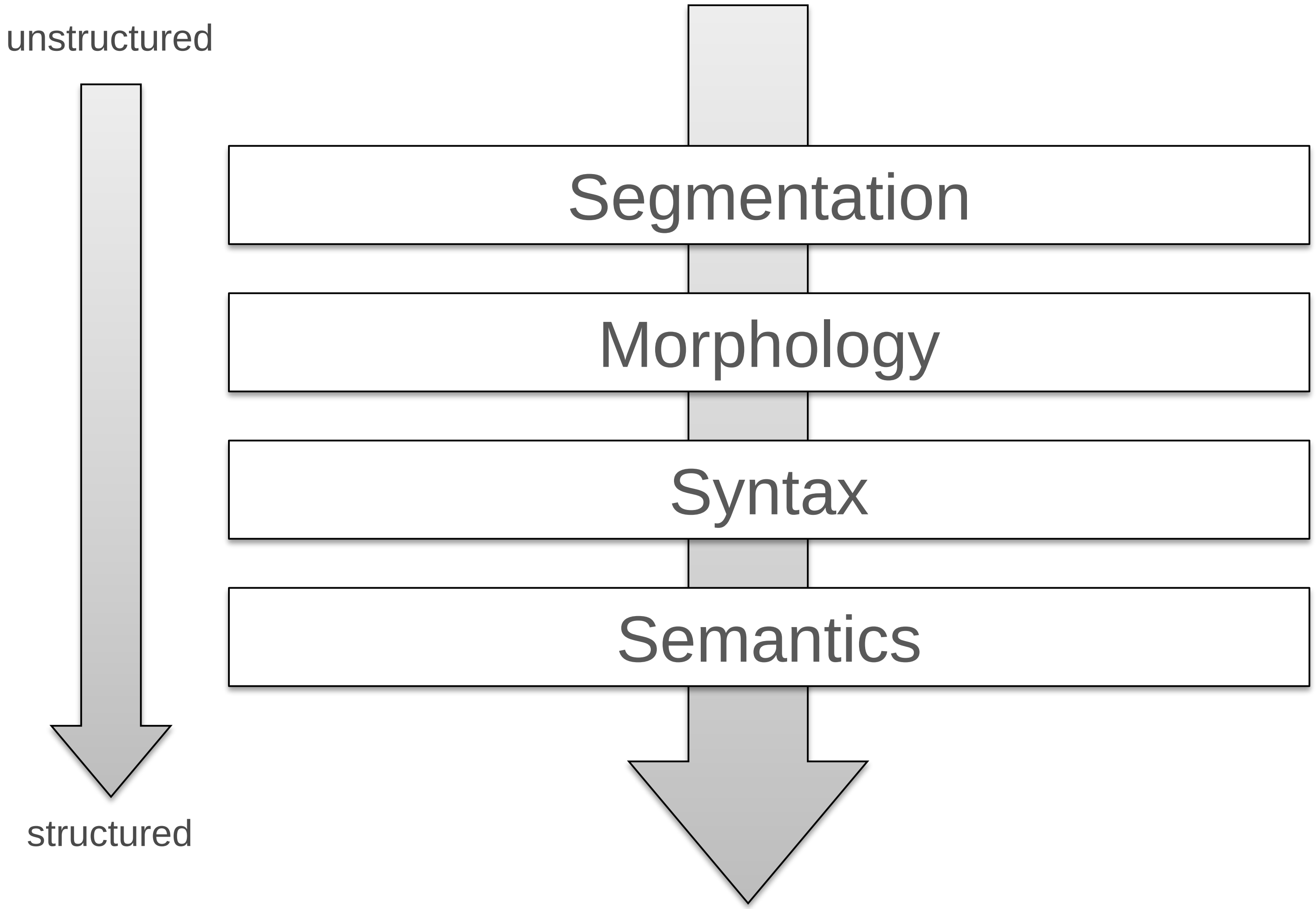
Introduction to DKPro Core

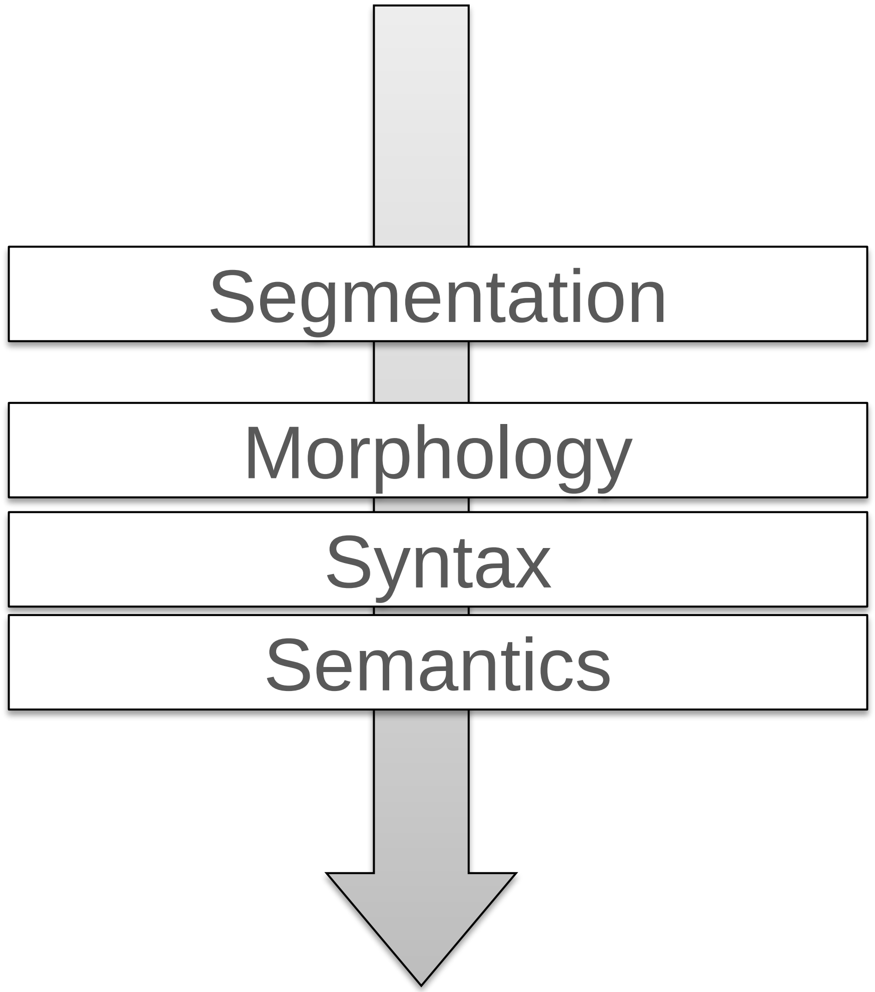
Dr. Judith Eckle-Kohler, Richard Eckart de Castilho, Roland Kluge, Dr. Torsten Zesch

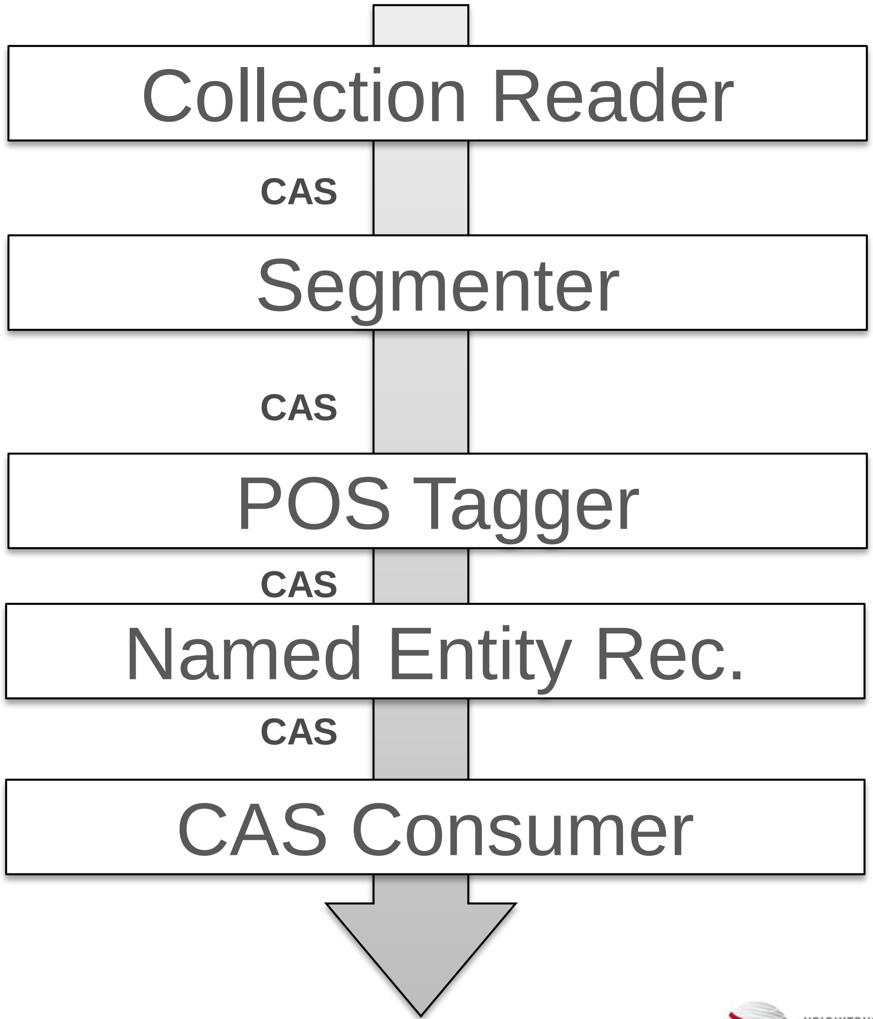
Part 1: Tools in DKPro-Core

Analysis Levels in Text Processing



UIMA Example Pipeline for Text Processing





Overview of Tools and Formats

Integrated Tools

- TreeTagger
- OpenNLP
- Stanford NLP
- JWordSplitter
- Language Tool
- MaltParser
- …

Supported Formats

- Text
- PDF
- TEI XML, BNC XML
- Negra Export
- SQL Databases
- Google web1t n-grams
- …

See also: http://code.google.com/p/dkpro-core-asl/
(list of important ASL / GPL components)

Overview of Tools and Formats – Sources

- Javadocs created by the DKPro-Core ASL Jenkins:

  https://zoidberg.ukp.informatik.tu-darmstadt.de/jenkins/job/DKPro Core ASL
- Overview of models in DKPro Core:

  see http://code.google.com/p/dkpro-core-asl/wiki
- Checkout DKPro Core in eclipse and browse the modules
  - The test classes provide important information on how to use the components

DKPro Core Type System

- Why does DKPro Core specify UIMA types for linguistic annotations?
  - Convenient access to linguistic annotations

```
for (N noun : JCasUtil.select (jcas, N.class)) {
    …
}
```

- See graphical overview of the most important types:
  - http://code.google.com/p/dkpro-core-asl/wiki/TypeSystem
- Where to find the DKProType System in the code, i.e., in the dkpro-core-asl modules?
  - de.tudarmstadt.ukp.dkpro.core.api.*
  - *TypeName.java* and *TypeName_Type.java*

UIMA type mappings – example POS tags

- Tags mapped to UIMA types (configurable)
  - To be found in src/main/resources, files named *.map
- **Generic:** Original tags stored in a *value* feature, e.g. POS.value
- **Coarse Grained:** Currently supported for Part-of-Speech tags
  - 13 coarse grained part-of-speech tags
  - ADJ, ADV, ART, CARD, CONJ, N (NP, NN), O, PP, PR, V, PUNC
- Convenient coarse-grained processing across languages
- Similar "Universal Part-of-Speech" tag-set published @ LREC 2012
  - *Slav Petrov, Dipanjan Das and Ryan McDonald*
  - Defines mappings for 25 tagsets in 22 languages
  - Will be adopted for DKPro Core in the future

Use of managed dependencies

```
<dependencyManagement>
…
  <dependency>
        <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
        <artifactId>de.tudarmstadt.ukp.dkpro.core-asl</artifactId>
        <version>1.4.0</version>
        <type>pom</type>
        <scope>import</scope>
  </dependency>
…
</dependencyManagement>
```

Readers for many formats

- Where to find readers in dkpro-core?
  - de.tudarmstadt.ukp.dkpro.core.io.*

- Example TextReader
  - Good to know: use this as a template, if you need to implement your own reader for any specific format

```
<dependency>
        <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
        <artifactId>de.tudarmstadt.ukp.dkpro.core.io.text-asl</artifactId>
</dependency>
```

Adding Models as managed dependencies, e.g. TreeTagger component

```
 <dependency>
      <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
      <artifactId>de.tudarmstadt.ukp.dkpro.core.treetagger-asl</artifactId>
 </dependency>

  <dependency>
      <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
      <artifactId>de.tudarmstadt.ukp.dkpro.core.treetagger-bin</artifactId>
  </dependency>
  <dependency>
      <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
      <artifactId>de.tudarmstadt.ukp.dkpro.core.treetagger-model-de</artifactId>
  </dependency>

<dependencyManagement>

...
 <dependency>
       <groupId>de.tudarmstadt.ukp.dkpro.core</groupId>
       <artifactId>de.tudarmstadt.ukp.dkpro.core.treetagger-asl</artifactId>
       <version>1.4.0</version>
       <type>pom</type>
       <scope>import</scope>
 </dependency>

...
</dependencyManagement>
```

Part 2: Linguistic annotation – Basics

Tokenization and sentence splitting – Ambiguities

Period

- In most of the cases: Final sentence punctuation symbol
- Part of an abbreviation, e.g. F.D.P.
- Numbers, ordinal numbers, e.g.: 21., numbers with fractions, e.g. 1.543
- References to resources locators, e.g.: www.apple.com
- To complicate things, if a sentence ends with an abbreviation which ends with a period, only one period is written. "He lives at Lakeview Dr."
- …

Whitespace character

- Part of numbers, e.g. "1 543"
- No segmentation character in multi-word expressions "New York"

Tokenization and sentence splitting – Ambiguities

Comma

- Part of numbers, e.g. 1,543

Single quote

- Within tokens to mark contractions and elisions, e.g. English: *don’t, won’t, you’ve, James’ new hat*; German: *Ich hab’s!*
- Part of a token in French, e.g. *aujourd´hui*
- But in **most cases:** Enclosing quoted groups of words

Dash

- A delimiter, if it connects strings of digits, e.g. "see page 100-101”
- In French: Signal a close connection between two tokens, e.g. verb and personal pronoun: *donne-le*
- In **most cases**, however, it is part of the token, e.g. *multi-word*

Morphology – Stemming

- Strip off the endings of words
  - sitting → sitt
- Stems do not necessarily correspond to a genuine word form
- Usually rule-based, no dictionary needed, excellent coverage

- Under-stemming
  - adhere → adher
  - adhesion → adhes
- Over-stemming
  - appendicitis → append
  - append → append

Morphology – Lemmatization

- “undo” the inflectional changes which a base form undergoes

cats	→ cat	

- Usually combined with part-of-speech tagging

left	→ leave	(verlassen/lassen)
left	→ left	(links)

- Has to deal with irregularities

sing, sang, sung	→ sing
indices	→ index
Bäume	→ Baum

Morphology – Stemming vs. Lemmatization

Original	Stemmed	Lemmatized
visibilities	visibl	visibility
adhere	adher	adhere
adhesion	adhes	adhesion
appendicitis	append	appendicitis
oxen	oxen	ox
indices	indic	index
swum	swum	swim

Morpho-Syntax – Part-of-Speech Tagger

- Assign grammatical category to tokens
  - Noun, verb, adjective, determiner, preposition, pronoun, …

- Sequence tagging model trained on a manually annotated corpus
  - Good to know: if possible, use exactly the same tokenizer that has been used to tokenize the training corpus for the tagger component
- Quality/coverage depends on training corpus
- Fall back rules
  - Suffix-based (-ion, -ly, …)
  - Numeric
  - Punctuation
  - …

Syntax – Chunker

A chunker annotates chunks

- To annotate chunks is partial parsing
- To annotate phrases is full parsing

Questions:

- What exactly is a chunk?
- What is the difference between chunks and phrases?

Understanding chunks requires understanding phrases.

Phrases

- **Phrase**: A group of words functioning as a single unit in the syntax of a sentence

- The central word defining the type (or syntactic category) of a phrase is called **head of the phrase.**
  - For a noun phrase, the head is the noun (or pronoun)

- Phrases are used in Phrase Structure Grammars.
- Constituency Parsing is based on Phrase Structure Grammars.
  - Constituents are phrases

Constituency Tests

Constituents can be identified using standard linguistic tests.

Example: The dog ate a cookie

- Substitution
  - The dog ate it
- Movement
  - A cookie was eaten by the dog
- Coordination with a constituent of the same phase type
  - The dog ate a cookie and a sausage
- Question

  What did the dog eat? A cookie

Phrases Types

- Phrases are classified by the type of head
  - **Prepositional phrase (PP)** with a preposition as head
    - e.g. from London, over the rainbow
  - **Noun phrase (NP)** with a noun as head
    - e.g. the black cat, a cat on the mat
  - **Verb phrase (VP)** with a verb as head
    - e.g. eat cheese, jump up and down
  - **Adjectival phrase (AP)** with an adjective as head
    - e.g. full of toys, very happy
  - **Adverbial phrase (AdvP)** with an adverb as head
    - e.g. very carefully

Heads and modifiers

- The **head** is the word which determines the syntactic type of the phrase
  - For a noun phrase, the head is the noun (or pronoun)

- **Modifiers** qualify another word or phrase
  - examples of modifiers are adjectives, adverbs, prepositional phrases

  all flights tomorrow(adverb)

  all flights from Cleveland (prepositional phrase)

- **Premodifiers** occur before the head

- **Postmodifiers** occur after the head

  all flights from Cleveland (prepositional phrase)

What is a chunk

Chunks are non-overlapping regions of text:

- (Usually) each chunk contains a head, with the possible addition of some preceding function words and modifiers
- Chunks are non-recursive:
  - A chunk cannot contain another chunk of the same category
- Chunks are non-exhaustive
  - Some words in a sentence may not be grouped into a chunk

Chunks vs Phrases

- Chunks are typically subsequences of constituents (they don't cross constituent boundaries)
  - noun chunks: everything in NP up to and including the head noun
    - NP the black cat on the tree -> noun chunk: the black cat
  - verb chunks: everything in VP (including auxiliaries) up to and including the head verb

Questions

- What are the basic steps for creating a DKPro-Core reader?
- How is the DKPro-Core type hierarchy organized?
- When to use DKPro-Core types?
- When are fine-grained POS tags needed? Give examples
- Where are models and resources stored (in DKPro-Core pipelines)?
- How to add models (e.g. tagger models, parser models) to your project?
- How do I access a corpus from DKPro?

Exercises (I)

Look at the example pipeline:

Run the example pipeline with different configurations

- Inspect lemmatization results
- Inspect chunks, discuss the limitations of chunking
- Inspect POS tags of verbs, discuss applications where the original POS tag is required (rather than the DKPro POS tag)

Exercises (II)

Adapt the example pipeline and write your own Consumers:

- Write a Consumer that identifies sentences with two consecutive noun chunks and no token tagged V in between
  - Inspect the annotation result, discuss

- Adapt the linguistic annotation pipeline to English
  - Experiment with two PDF files from the educational domain:
    - `src/main/resources`
  - Adapt the reader and the tagger accordingly

References

- Steven Abney. Parsing By Chunks. In: Robert Berwick, Steven Abney and Carol Tenny (eds.), *Principle-Based Parsing*. Kluwer Academic Publishers, Dordrecht. 1991.