ISMLA Session 4 - UIMA

Björn Rudzewitz

Tübingen University

November 13, 2017

Plan

1. Collection Reader
2. CPE/CPM
3. CASConsumer
4. Use Cases

UIMA Workflow

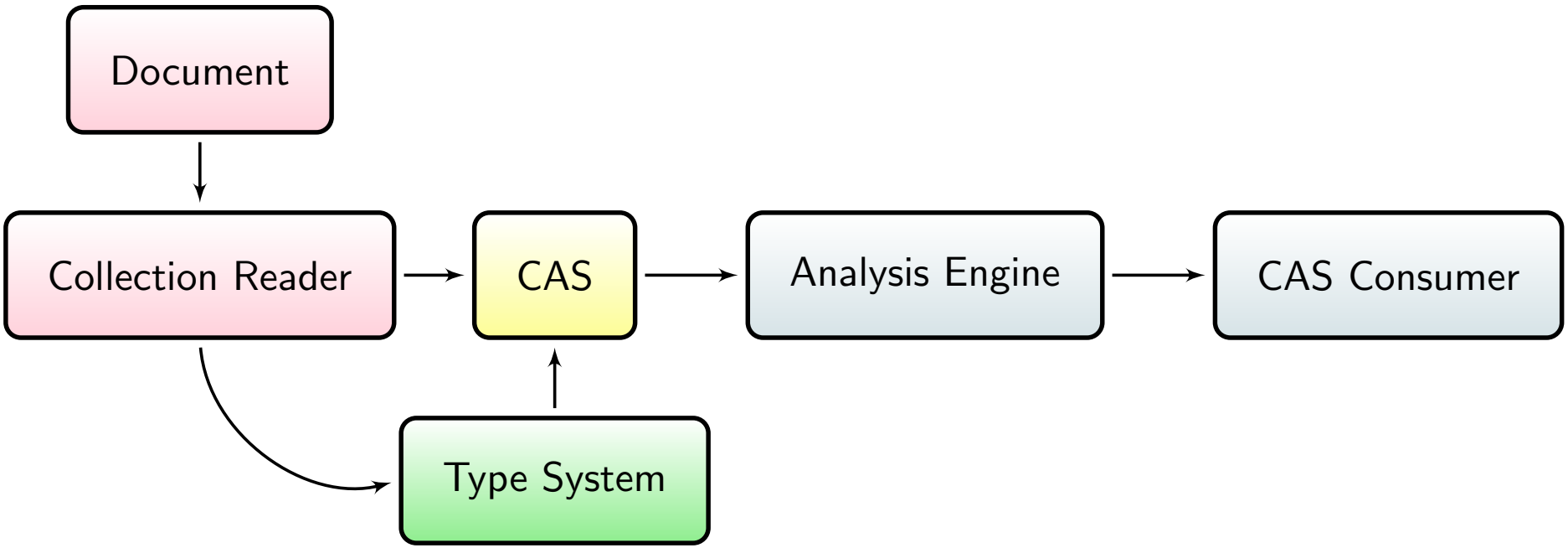

Figure: UIMA Workflow

Exercise

last session:

- writing analysis engine wrapping OpenNLP tokenizer
- tokenizer should receive sentences as input
- writing AAE chaining sentence detector and tokenizer
- test via DocumentAnalyzer

Exercise

- most of the code exactly like in sentence detector (initialization, loading tool, obtaining spans, . . .)
- one pitfall: setting token indices

Exercise

- most of the code exactly like in sentence detector (initialization, loading tool, obtaining spans, . . .)
- one pitfall: setting token indices
  - individual sentences as input to tokenizer
  - tokenizer computes spans relative to sentence
  - token indices need to be set relative to the document, not only sentence

```
token.setBegin(sentence.getBegin() + token.getBegin())
```

Collection Reader

- initializes for each document a Common Analysis Structure
- observations can be extracted from various sources, e.g.
  - a CSV file,
  - a directory with files,
  - a web resource, ...
- initializes an iterator over observations, then while processing selects the next element
- pipeline automatically stops when iterator has no next element
- essentially a collection reader assumes a specific file format and creates an empty container for subsequent analyses

Collection Reader

- associates a document text and language with every document (i.e. entity to be processed)
- provides the framework with material to be processed and progress information
- before AEs can be called, a collection reader sets up a CAS for every document
- similar design pattern like `java.util.Iterator` with `getNext` and `hasNext` functions

Collection Reader

- extends `org.apache.uima.collection.CollectionReader_ImplBase`
- functions:
  - getNext: sets language and document text
  - hasNext
  - getProgress: possibility to indicate progress in different measures
  - close
  - (initialize): prepares component for iterating
- Java code and descriptor file (*New* → *Other* → *UIMA* → *Collection Processing Components* → *Collection Reader Descriptor File*)

Collection Reader

- for writing output, sometimes the original file name needs to be remembered
- e.g DocumentAnalyzer creates output files *"'originalName.xmi"*
- solution: create a type `DocumentMetaData` in the Collection Reader to store global meta data about a document
- the DocumentAnalyzer also creates a type `DocumentAnnotation` with the language of the original document

CPM

testing/applying a Collection Reader:

- DocumentAnalyzer predefines a Collection Reader (reads all files in a directory)
- more powerful variant: **CPM Frame**
  - allows to define (in addition to AEs) also Collection Reader and CAS Consumer
  - allows to define "source-to-sink"[1] UIMA pipelines

[1] `https://uima.apache.org/d/uimaj-current/overview_and_setup.html#ugr.ovv.conceptual.applicaiton_building_and_collection_processing`

Collection Processing Engine

- Collection Processing Engine (CPE) to manage the complete flow of a UIMA application
- defines
  1. collection reader
  2. analysis engine(s)
  3. CAS consumer(s)
- CPM Frame allows to create, save, load CPE descriptors

CPM

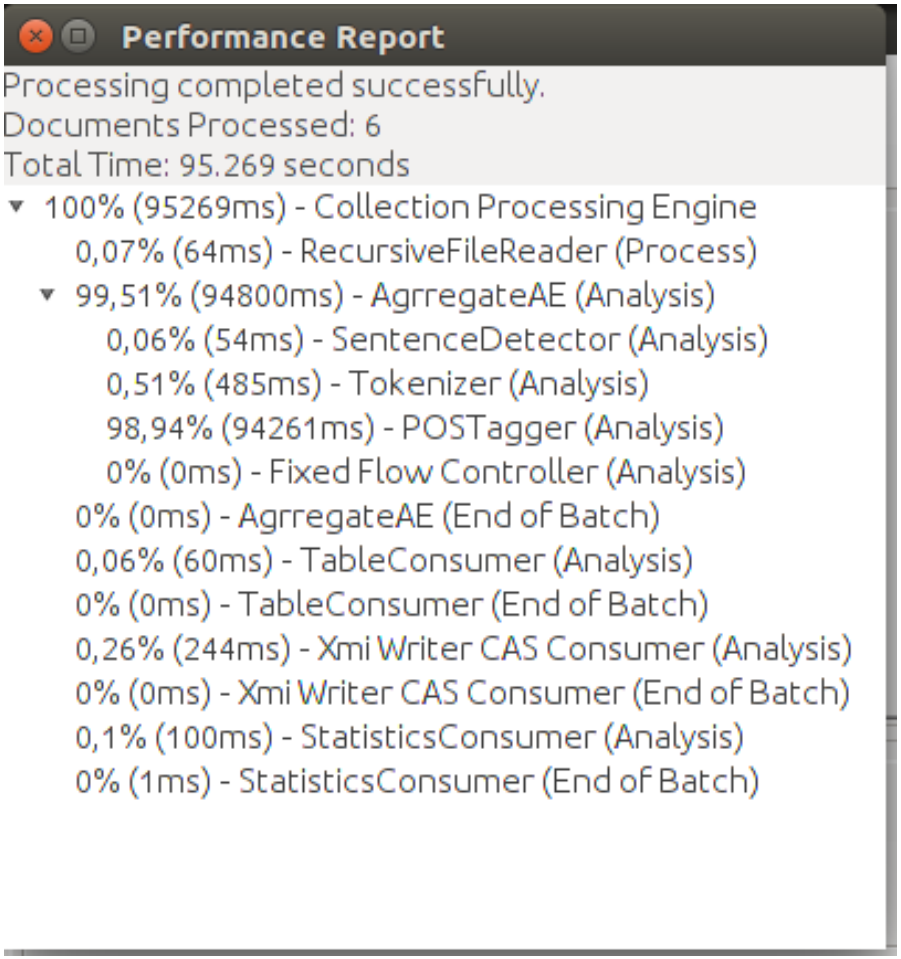

Figure: CPM Performance report after completing processing

Demonstration

- writing a collection reader to recursively read all files

Exercise

- write a collection reader that reads in every line of a file as a document (see last session's handout, exercise 2)

CASConsumer

- final component in pipeline
- during processing: CAS in memory, but not available outside the program
- CASConsumer outputs the results of the analyses for other applications
- Example: write results to a data base, a CSV file, XMI collection
- *"typically aggregate the document-level analyses in an application-dependent data structure"*[Ferrucci and Lally, 2004, p. 333]
- provides options for error handling, performance monitoring, parallelization

CASConsumer

functions:

- `processCAS`: output information from the CAS' meta data indices
- (`initialize`): often used to check if output resource is available, open global output file, etc.
- (`destroy`): called at the very end of the process, useful for closing files

CASConsumer

Demonstration

- write a CASConsumer that outputs a table with word and POS tag separated by tabs

CASConsumer

Exercise

- write a CASConsumer that for each file outputs the
  - source path
  - document language
  - number of sentences
  - number of tokens
  - number of token types
  - type-token-ratio
- test the CASConsumer via the CPMFrame

Example Use Cases

- UIMA being an "empty" framework, its application is flexible and has been adapted to a range of tasks
- scalability of framework to large data volumes supports usage in research and industry

Example Use Cases

- Multilingual NLP
- Question answering, e.g. IBM Watson [Ferrucci et al., 2010]
- Information extraction, e.g. [Savova et al., 2010]
- Taxonomy extraction, e.g. [Gates et al., 2005]
- Complexity analysis, e.g. [Chen and Meurers, 2016]
- ...

Xiaobin Chen and Detmar Meurers. Ctap: A web-based tool supporting automatic complexity analysis. *CL4LC 2016*, page 113, 2016.

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.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79, 2010.

Stephen C Gates, Wilfried Teiken, and Keh-Shin F Cheng. Taxonomies by the numbers: building high-performance taxonomies. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 568–577. ACM, 2005.

Guergana K Savova, James J Masanz, Philip V Ogren, Jiaping Zheng, Sunghwan Sohn, Karin C Kipper-Schuler, and Christopher G Chute. Mayo clinical text analysis and knowledge extraction system (ctakes): architecture, component evaluation and applications. *Journal of the American Medical Informatics Association*, 17(5):507–513, 2010.