Apache UIMA and Mayo cTAKES

UIMA and how it is used in the clinical domain

Lars-Erik Bruce

March 16, 2012

Outline

Outline

What is UIMA?
(You - eee - muh)

- Unstructured Information Management Architecture.
- Open source scaleable and extensible platform.
- Create, integrate and deploy unstructured information management solutions.
- Several Open Source projects based on UIMA

Why UIMA?

Many NLP projects have the same work-flow:

- Read a number of documents from a collection.
- Run a number of NLP algorithms in a specific order on each document.
- Use the result to:
  - Train a model.
  - Classify documents or parts of documents.
  - Find named entities
  - Store and/or visualize the findings.
  - etc ...

What is UIMA?

If developers of NLP tools where to agree on a mutual standard for managing the work-flow and information exchange between components, it would be easier to share and re-use resources. IBM developed UIMA for reasons as these.

- Precisely controlled work flow.
- Good scalability abilities.
- Easy to utilize modules created by third party developers.
- Fairly easy to wrap 3rd party tools as new components.
- Ongoing development on new resources.

UIMA key facts

- Open Source Java Framework
- Developed by IBM, now maintained by Apache.
- Principle infrastructure in DeepQA and Watson.
- Supported and used by DARPA.
- Standardized at OASIS.

UIMA Components and pipeline modules

The main components of UIMA

- The Common Analysis System/Structure (CAS)
- The Type System
- The Collection Processing Manager

The types of modules in the pipeline

- Collection reader
- Analysis Engine
- CAS consumer

The Common Analysis System/Structure (CAS)

- Provides an interface for storing and retrieving information.
- One CAS-object per document.
- Serializable, can be sent across virtual and physical machines.

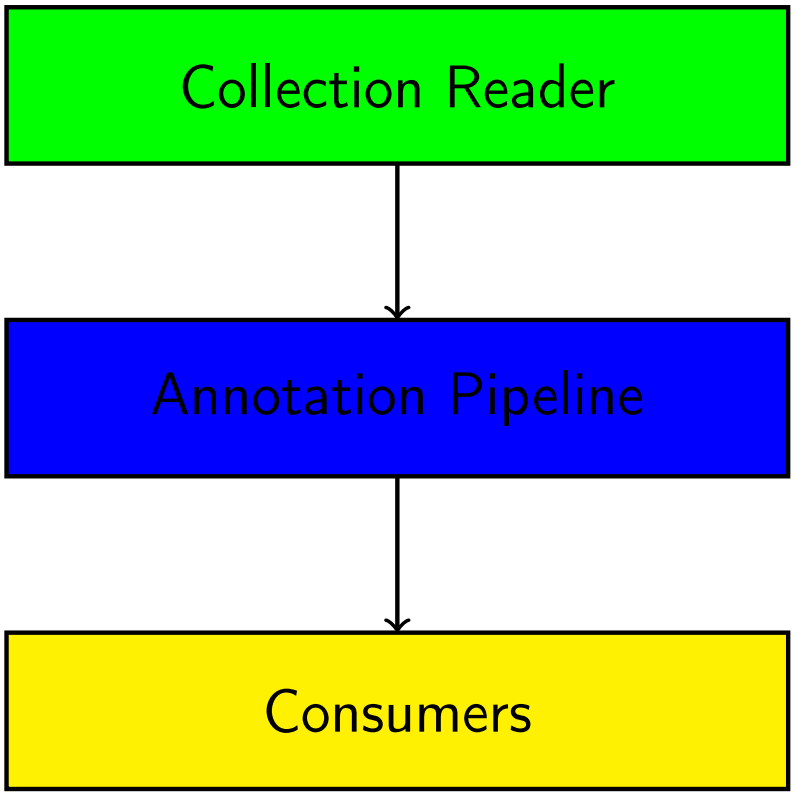
Type system

- Defines the output of the Analysis Engines.
- Typed Feature Structures.
- Name spaces for each type, to avoid name conflicts.

Actually we can assign "types" that should be used per annotator and collection reader. The system of types (type system) is created when the pipeline is started.

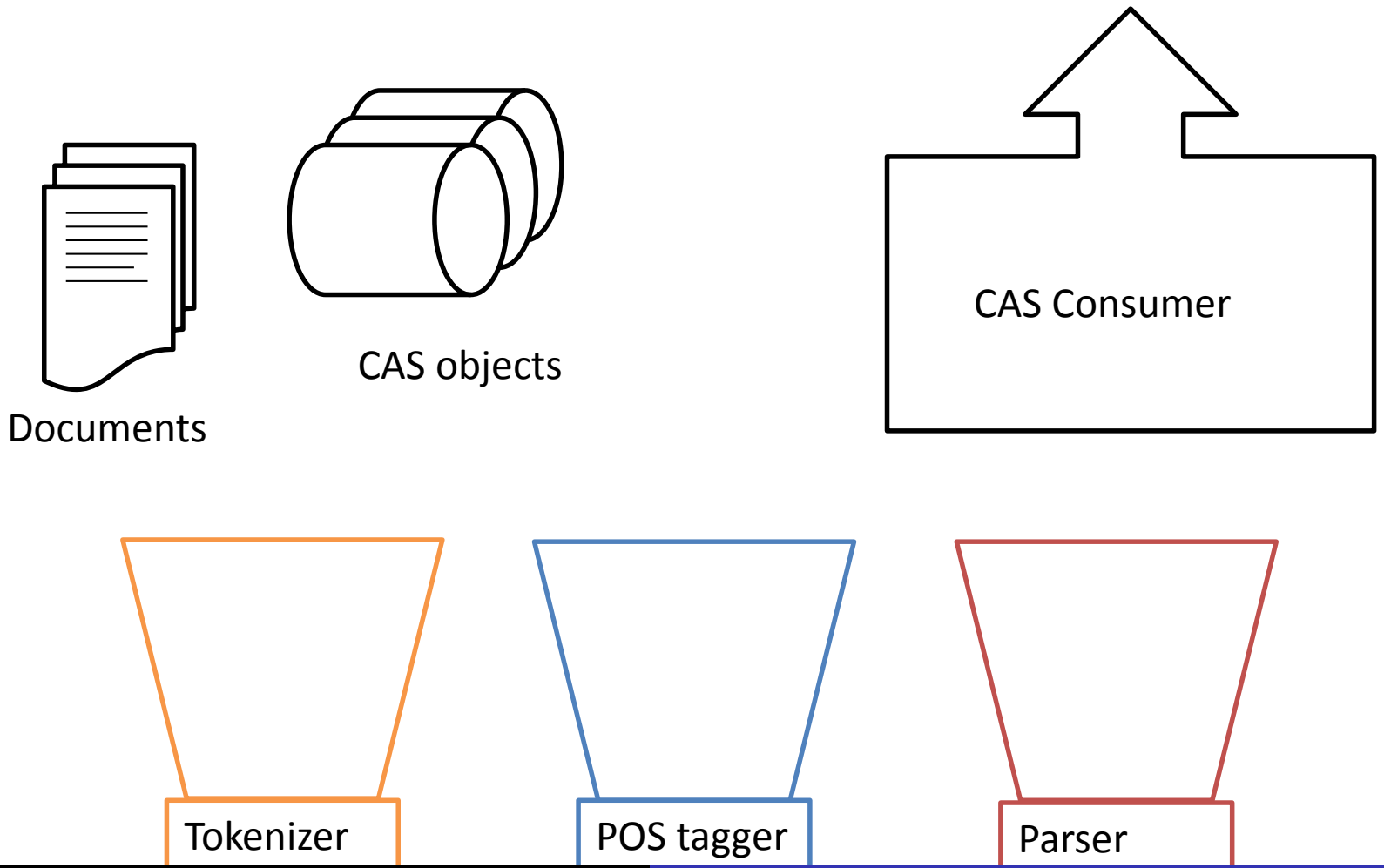
$$Patient \left[\begin{array}{ll} Name & JohnJoe \\ BirthDate & 02.02.62 \\ Allergies & \left\langle \left[\; Drug \quad 'penicillin' \; \right], \ldots \right\rangle \\ Diagnoses & \left\langle \left[\; Illnes \quad 'diabetes', \ldots \; \right], \ldots \right\rangle \end{array} \right]$$

Collection Processing Engine/Manager/Architecture

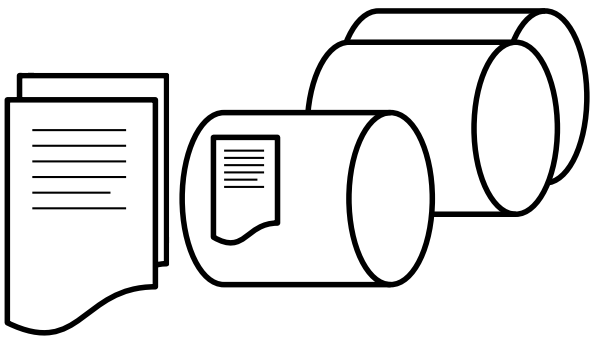


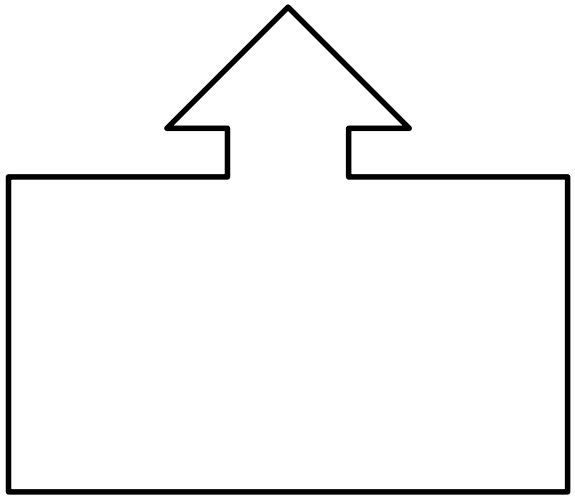
- **Engine** describes which modules to use.
- **Manager** reads the description, making it happen.
- **Architecture** defines components and interfaces.

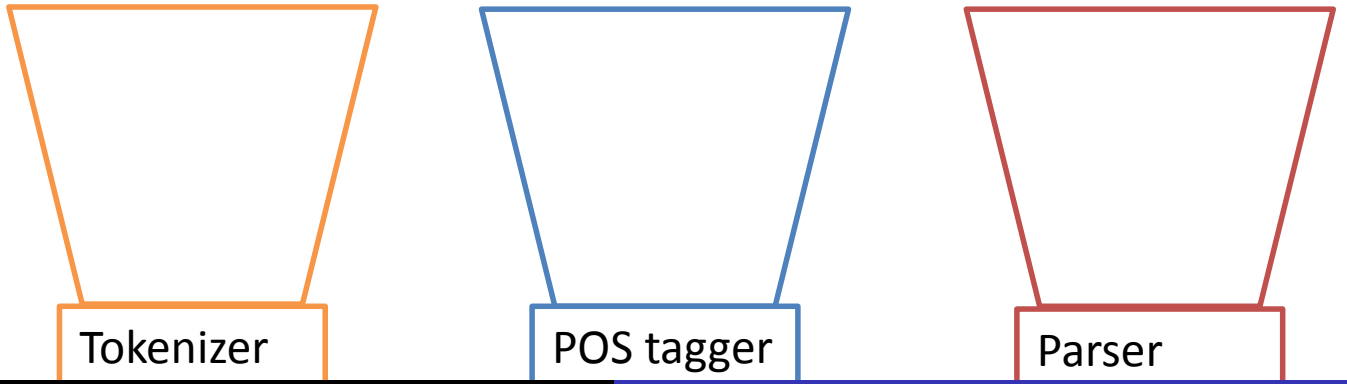
The lifetime of a pipeline



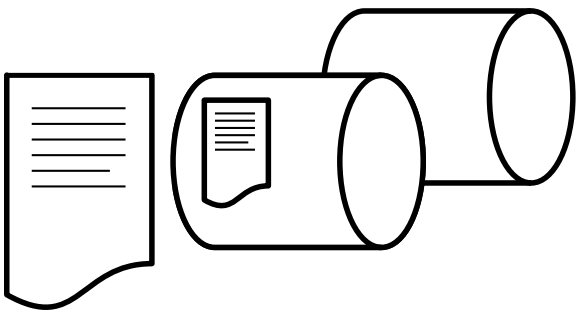
The lifetime of a pipeline

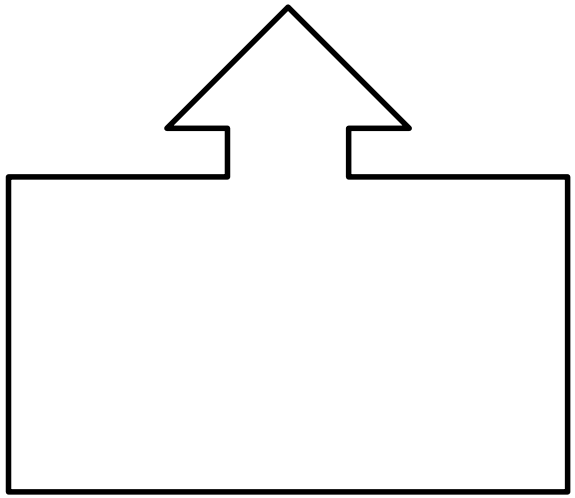


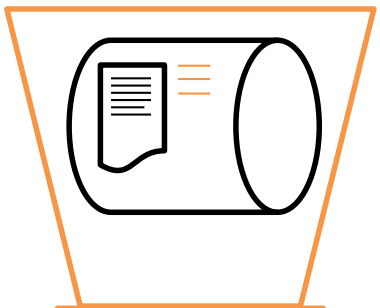


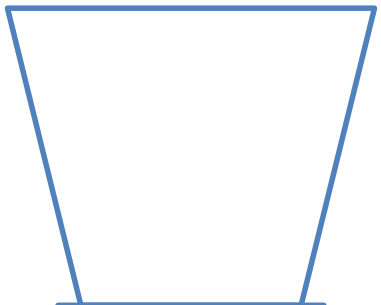


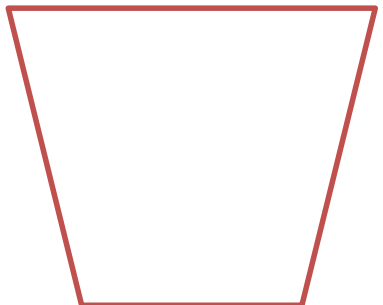
The lifetime of a pipeline









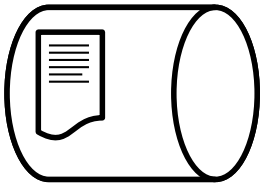


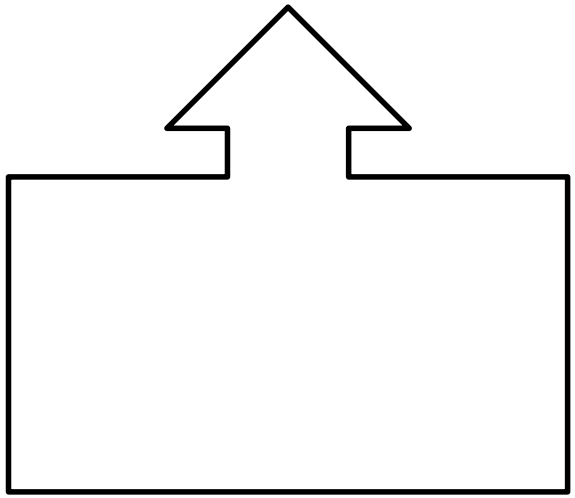
Tokenizer

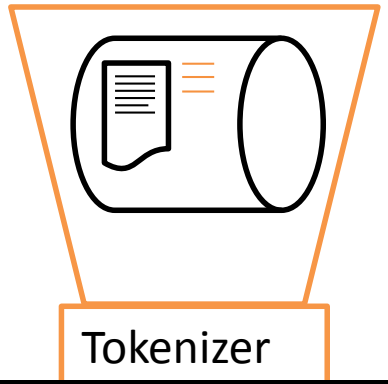
POS tagger

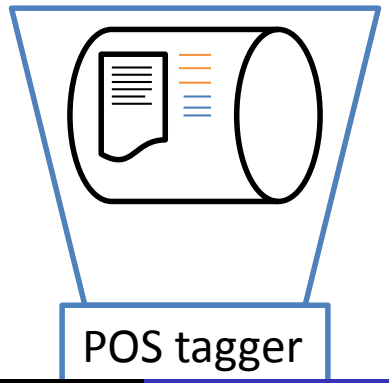
Parser

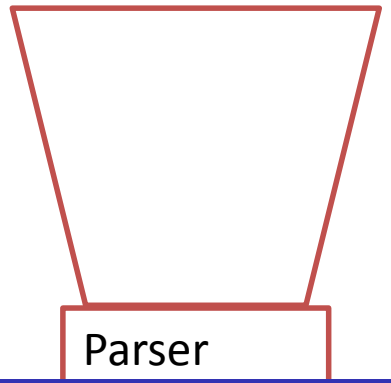
The lifetime of a pipeline



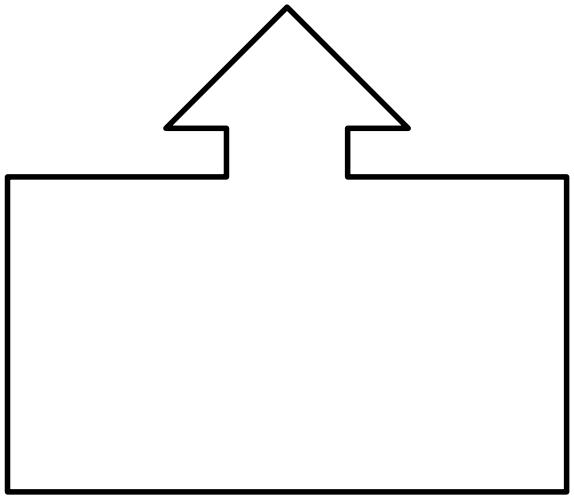


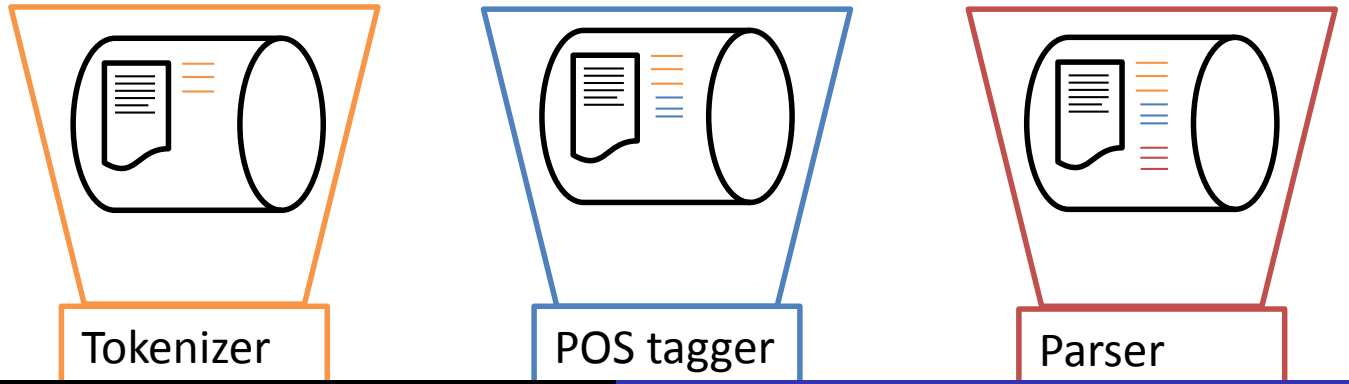




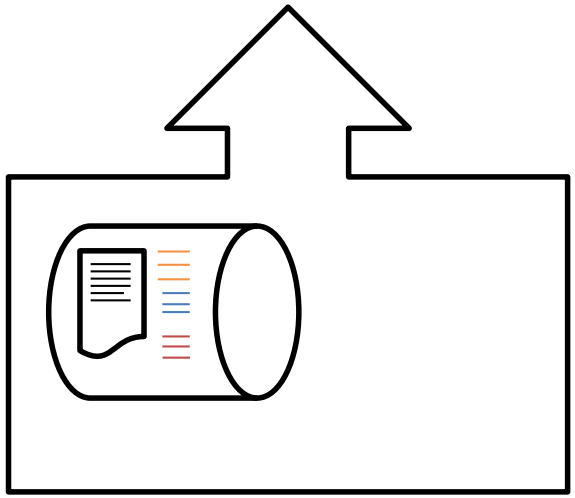


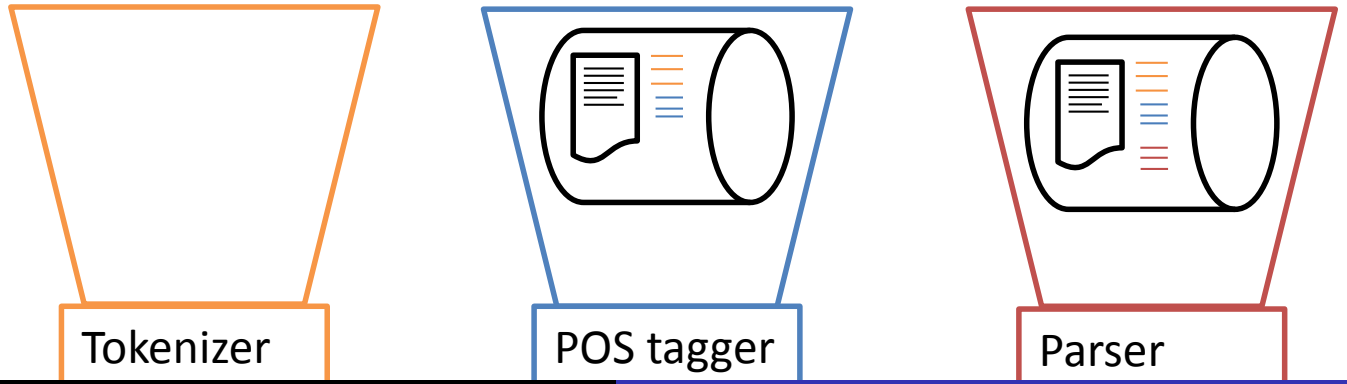
The lifetime of a pipeline



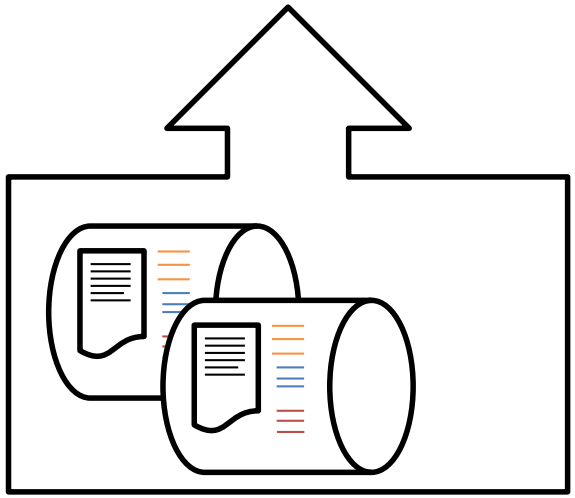


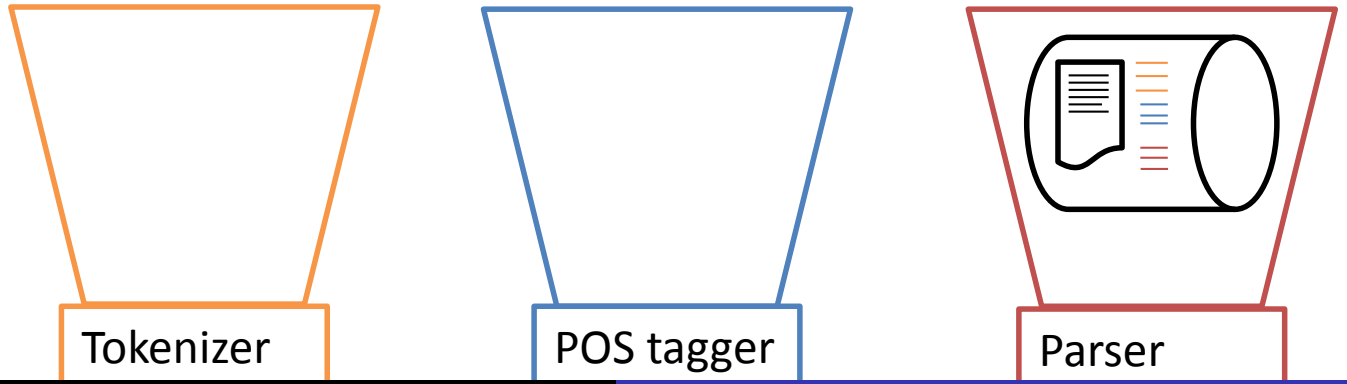
The lifetime of a pipeline



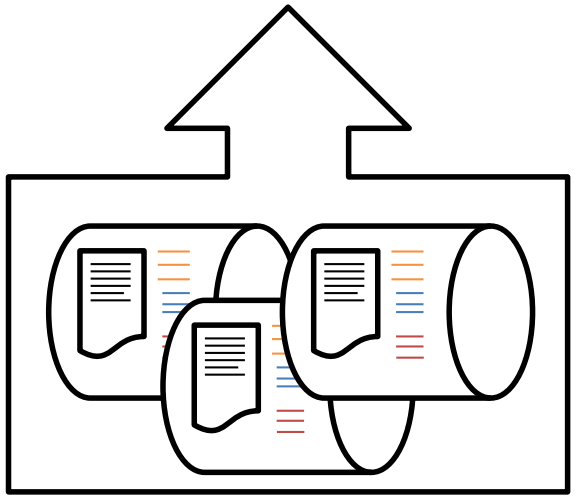


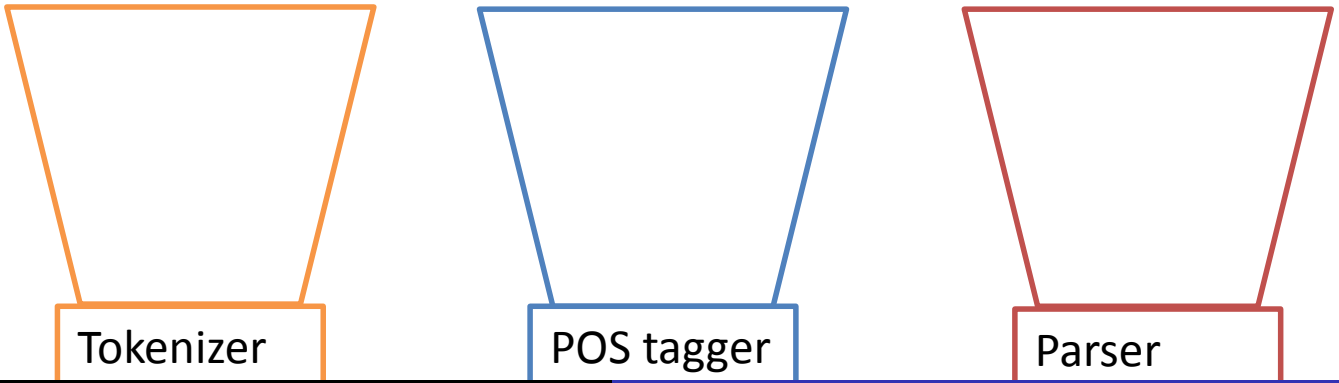
The lifetime of a pipeline





The lifetime of a pipeline





Collection Reader

- Ensures uniform access to the data.
- Knows how to iterate over input documents.
- Stores the raw data to the CAS Object.
- The data is now available to the analysis pipeline.

Collection Reader
Implementation

- We must describe the collection reader
  - Which parameters should the user set.
  - What Type System should be used.
- and write the actual code
  - *initialize()* reads the user input parameters.
  - *getNext(CAS)* inserts a document to the CAS object.
  - *hasNext()* returns true if there are more documents to read.
  - *close()* closes open resources.

Analysis Engine

- Analyses artefacts (documents) and infers new information.
- Reads raw-data of the artefact from CAS objects.
- Retrieves previous stored information for further analysis.
- Stores new inferred information to CAS objects.

Analysis Engine
Implementation

- We must describe the analysis engine
  - Which parameters should the user set.
  - What Type System should be used.
- and write the actual code
  - *process(CAS)* analyses the document and inserts new findings in the CAS.

This is really all you need. Typically an AE is just a wrapper for other tools, like for instance OpenNLP or Weka.

CAS Consumer: The CAShing in

- Iterates over each CAS-object when all analysis is done.
- Defines the end-point of the extracted information:
  - Databases
  - Feature extraction/machine learning
  - Index for search engines

CAS Consumer
Implementation

- We must describe the CAS consumer ...
  - Which parameters should the user set
  - What Type System should be used
- and write the actual code
  - *process(CAS)* processes the CAS-object, just like an AE
  - *collectionProcessComplete()* is called when all documents are processed by AEs.

cTAKES

Mayo clinical Text Analysis and Knowledge Extraction System

- Developed by the Mayo clinic and IBM
- Released as part of the Open Health Natural Language Processing Consortium
- Processes a collection of patient records to recover:
  - Tokens, titles, measurements
  - POS tags and chunks
  - Word dependencies and constituencies
  - Named entities and negations

The task: Reading and analyzing the I2B2 2008 corpus in cTAKES

Let us introduce the cTAKES system by see how we can use it to analyze the I2B2 2008 shared task corpus.

- A corpus of electronic discharge summaries, patients with obesity and 15 co-morbidities.
- Multi-class, multi-label, classification task.
- Annotations of morbidities at document level, no other annotations.
- Pure text and a "problem": Line shifts inserted in text wraps.

cTAKES Collection Reader

- Supports plain text and "Clinical Document Architecture" (CDA)
- We need to build a Collection Reader for the I2B2 corpora.
- First we must create the descriptor:

Eclipse show

Implementing the Collection Reader

This is fairly easy. We traverse the DOM-tree of the XML-file the documents are in, iterating over the doc-elements and inserting text and document id into the CAS object.

cTAKES Analysis Engines

A bunch of features

- Tokenizer
- Context dependent tokenizer
- Sentence Detector
- Normalizer, wrapper for the SPECIALIST lexical tools
- POS tagger
- Chunker
- Dependency Parser
- Dictionary lookup
- Context annotator
- Constituency parser

Let us run the following record excerpt through the pipeline

"The patient did not feel any chest pain,
but Dr. Joe confirmed myocardial infarction."

.

cTAKES Tokenizer

- Consists of two parts:
  - Simple splitter (space and punctuation)
  - FSA which merges tokens to create dates, fractions, titles, roman numerals, etc.

The patient did not feel any chest pain, ↩
but Dr. Joe confirmed myocardial infarction.

- WordToken
- PunctuationToken
- NewlineToken
- PersonTitle
- RomanNumeral

cTAKES Sentence detector

- Wrapper around OpenNLP supervised ME sentence detector tool.
- Sentence annotations based on location of end-of-line character and output of OpenNLP annotator.
- Trained on annotated clinical data.
- The annotated material is not available, so we have to stick with this model.

... it. The patient did not feel any chest pain,↩
but Dr. Joe confirmed myocardial infarction. With ...

- Sentence

cTAKES Normalizer (SPECIALIST Lexical Variant Generation)

- Wrapper around the SPECIALIST Lexical Tool component "norm"
- Produce alternative tokens with respect to
  - Alphabetic case
  - inflection
  - Spelling variants
  - Punctuation
  - Genitive markers

cTAKES Normalizer (SPECIALIST Lexical Variant Generation)

Normalizer output on the token "patient"

$$
\begin{bmatrix}
\textit{FSList} & \left[\begin{array}{lll} \textit{head} & \textit{Lemma} & \left[\begin{array}{ll} \textit{key} & \text{“}\textit{patients}\text{”} \\ \textit{posTag} & \text{“}\textit{NN}\text{”} \end{array}\right] \end{array}\right] \\
\textit{tail} & \textit{FSList} & \left[\begin{array}{lll} \textit{head} & \textit{Lemma} & \left[\begin{array}{ll} \textit{key} & \text{“}\textit{patient}\text{”} \\ \textit{posTag} & \text{“}\textit{JJ}\text{”} \end{array}\right] \\ \textit{tail} & \textit{FSList} & \left[\begin{array}{ll} \textit{head} & \textit{Lemma}\,[\ldots] \\ \textit{tail} & \textit{EmptyFSList} \end{array}\right] \end{array}\right]
\end{bmatrix}
$$

cTAKES POS tagger and chunker

- The POS tagger and chunker are also wrappers around the OpenNLP modules.
- cTAKES provides models trained on clinical data.
- And again, the training data is not available.

The	*patient*	*did*	*not*	*feel*	*any*	*chest*	*pain*	,	$\hookleftarrow$
DT	*NN*	*VBD*	*RB*	*VB*	*DT*	*NN*	*NN*	,	*NN*

($_{NP}$The patient) ($_{VP}$did not feel) ($_{NP}$ any chest pain,($_{NP}$ $\hookleftarrow$)
but ($_{NP}$Dr.) ($_{NP}$Joe)) ($_{VP}$confirmed) ($_{NP}$myocardial infarction).

cTAKES Dictionary lookup

Composed of several components

- Lookup window annotation
  - Marks all NP phrases as lookup windows.
  - If lookup-window A subsumes lookup-window B, delete B.
- Dictionary lookup annotation
  - Use the token and all expanded versions to compose a dictionary query
  - Collects concepts from UMLS (SNOMED and RxNorms)

The patient did not feel **any chest pain,**↩ **but Dr. Joe** confirmed **myocardial infarction**.

Dictionary lookup

"Myocardial infarction" as a named entity

- Dictionary lookup finds six SNOMED-CT concepts for the lookup-window "myocardial infarction".
- All of those are marked "ambiguous" in the terminology, except from one:

Concept	Myocardial infarction
ID	D3-15000
Status	Current
Descriptin	Myocardian infarction (disorder)

Context annotator

- Negation detection based on the NegEx algorithm
- Searches for negation clues to the left and right of discovered entities
- Discovers the term “any” to the left of “chest pain”, “chest” and “pain”
- Inserts feature “certainty” with value ’-1’ into each of these NEs

ContextAnnotation	
begin	25
end	28
FocusText	“Chest pain”
Scope	“LEFT”

References

- Götz, T., Suhre, O 2004 "Design and implementation of the UIMA Common Analysis System" *IBM Systems Journal*, IBM, Riverton
- Hahn, U. et al 2007 "An Annotation Type System for a Data-Driven NLP Pipeline", *Proceedings of the Linguistic Annotation Workshop*, Association for Computational Linguistics, Prague
- Savova, G. et al 2008 "UIMA-based Clinical Information Extraction System", *LREC 2008: Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP*
- Savova, G. et al 2010 "Mayo clinical Text Analysis and Knowledge Extraction System (cTAKES): architecture, component evaluation and applications", *JAMIA*, J Am Med Inform Assoc