Bluima: a UIMA-based NLP Toolkit for Neuroscience

UIMA GSCL-2013, Darmstadt, Germany

Renaud Richardet, Jean-Cédric Chappelier, Martin Telefont
Blue Brain Project, EPFL, Switzerland renaud.richardet@epfl.ch

Agenda

- Bluima, NLP for neuroscience
- Experiences with learning UIMA
- Scripting language to design UIMA pipelines
- MongDB CAS store

Speaker:

- Renaud Richardet, PhD candidate Neuroscience
- 10 years software dev., renaud@apache.org
- now: BioNLP at BlueBrain, EPFL, Switzerland

Bluima, NLP for Neuroscience

- BlueBrain's bioNLP toolkit, based on UIMA
- goal: extract structured data from PubMed to support researchers in brain modeling
- focus: extracting entities that are specific to neuroscience (like brain regions and neurons)
- examples:
  - Brain region x brain region
  - Protein concentration
  - Neuron properties

Before UIMA

- While research scientist at Fachhochschule Windisch, Switzerland
- Project: **semantic matching from job advertisement** (1Mio / year) **and resume** profiles (from PDF)
- Took 2 months to re-invent a CAS structure
- Another 2 to re-invent a RUTA-like DSL
- Wished I had used UIMA…

RUTA-like DSL (Pseudocode)

```
// Method calls
String DT = "( $isDegreeType() )";
// Regular expressions
String IM = " /in|im|als|mit|der|des|zu|zur|zum|\\(/";
String ODER = " /(\\/|,|und|oder|/)/";
// Annotations
String TYPES = "[#JobTitle|#EducationTitle]";


// A1: degree then edu
PatternMatcher A1_0 = parse(DT + IM + TYPES + ODER + TYPES + ODER + TYPES);
PatternMatcher A1_1 = parse(DT + IM + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_2 = parse(DT + " /i.*/ /de.*/" + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_3 = parse(DT + IM + TYPES + ODER + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_4 = parse(DT + IM + TYPES + ODER + TYPES + ODER + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_5 = parse(DT + IM + TYPES + ODER + TYPES + " * oder" + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_6 = parse(DT + IM + TYPES + " *" + TYPES).addIgnoreTag(PKT);
PatternMatcher A1_7 = parse(DT + IM + TYPES + ODER + TYPES + ODER + IM + TYPES);

// A2: edu then title
PatternMatcher A2_2 = parse(TYPES + ODER + TYPES + ODER + TYPES + " " + DT);
PatternMatcher A2_3 = parse(TYPES + ODER + TYPES + " " + DT);
PatternMatcher A2_6 = parse(TYPES + ODER + TYPES + " " + DT).addIgnoreTag(PKT);
PatternMatcher A2_7 = parse(TYPES + " " + DT).addIgnoreTag(PKT);
PatternMatcher A2_8 = parse(TYPES + " * " + DT).addIgnoreTag(PKT);
```

Learning UIMA

- Steep curve at first, too many acronyms, too much XML
- IBM & Apache: good pedigree, trust
- Very well thought concepts and API; solid implementation; never had to digg into UIMA's core library
- UIMAFit facilitates pipeline development
- FAIL: Pear → ended up providing a resource path, and package UIMA with Maven

Integrating Existing Modules into Bluima

- Typesystem and Tokenizer based on JULIELab modules
- Protein Named entity recognizers (NERs) (ABNER, BANNER, Gimli)
- Other NERs (OSCAR4, Linnaeus)
- Many lexical NERs with ConceptMapper

Lexica in Bluima

Name	Source	Scope	# forms
Age	BlueBrain	age of organism, developmental stage	138
Sex	BlueBrain	sex (male, female) and variants	10
Method	BlueBrain	experimental methods in neuroscience	43
Organism	BlueBrain	organisms used in neuroscience	121
Cell	BlueBrain	cell, sub-cell and region	862
Ion channel	Channelpedia [27]	ion channels	868
Uniprot	Uniprot [1]	genes and proteins	143,757
Biolexicon	Biolexicon [30]	unified lexicon of biomedical terms	2.2 Mio
Verbs	Biolexicon	verbs extracted from the Biolexicon	5,038
Cell ontology	OBO [2]	cell types (prokaryotic to mammalian)	3,564
Disease ont.	OBO [23]	human disease ontology	24,613
Protein ont.	OBO [20]	protein-related entities	29,198
Brain region	Neuronames [3]	hierarchy of brain regions	8,211
Wordnet	Wordnet [7]	general English	155,287
NIFSTD	NIF [12,4]	neuroscience ontology	16,896

Table 1. Lexica and ontologies used for lexical matching.

Integrating Existing Modules into Bluima

- Many modules and models available for BioNLP, jumpstarted
- Lego-like composition of pipelines is great
- Often, research-grade code; many & exotic dependencies; lots of duplication, every module does its own preprocessing (e.g. BANNER, OSCAR). But still much better that writing these modules ourselves!
- Did not abstract the typesystem yet

Pipeline Scripting Language

Tool	Advantages	Disadvantages
UIMA GUI	GUI	minimalistic UI, can not reuse pipelines
XML descriptor	typed (schema)	very verbose
raw UIMA java API	typed	verbose, requires writing and compiling Java
UIMAFit	compact, typed	requires writing and compiling Java code

Table 2. Different approaches to writing and running UIMA pipelines.

Pipeline Scripting Language

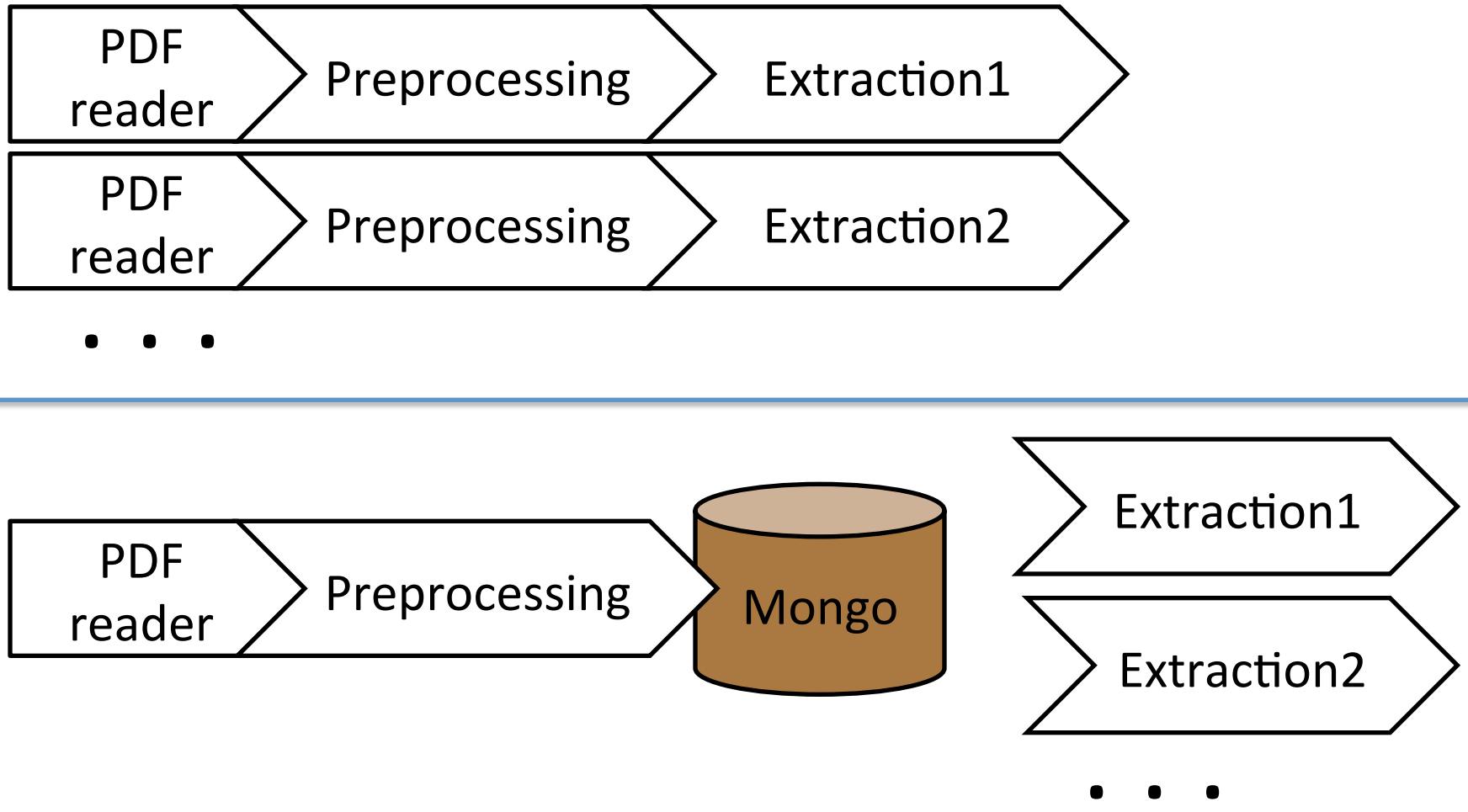
- A minimalistic scripting (domain-specific) language, allowing UIMA pipelines to be configured with text files, in a human-readable format
- Goals
  - Improve faster and more lightweight design and experimentation with UIMA pipelines
  - Enable researchers without Java or UIMA knowledge to easily design and run pipelines

Pipeline Script Example

```
# collection reader configured with a list of files (provided as external params)
cr: FromFilelistReader
 inputFile: $1
# processes the content of the PDFs
ae: ch.epfl.bbp.uima.pdf.cr.PdfCollectionAnnotator

# tokenization and lematization
ae: SentenceAnnotator
 modelFile: $ROOT/modules/julielab_opennlp/models/sentence/PennBio.bin.gz
ae: TokenAnnotator
 modelFile: $ROOT/modules/julielab_opennlp/models/token/Genia.bin.gz
ae: BlueBioLemmatizer

# lexical NERs, instantiated with some helper java code
ae_java: ch.epfl.bbp.uima.LexicaHelper.getConceptMapper("/bbp_onto/brainregion")
ae_java: ch.epfl.bbp.uima.LexicaHelper.getConceptMapper("/bams/bams")

# removes duplicate annotations and extracts collocated brainregion annotations
ae: DeduplicatorAnnotator
 annotationClass: ch.epfl.bbp.uima.types.BrainRegionDictTerm
ae: ExtractBrainregionsCoocurrences
 outputDirectory: $2
```

Table 3. Pipeline script for the extraction of brain regions mention co-occurrences from PDF documents.

MongoDB CAS Store



Existing CAS Stores

- XCAS, XMI
- ZipXCAS, ZipXMI
- BinaryCasReader (DKPro)
- NEW: Database (Fette, 2013)

MongoDB

- Popular, open source
- NoSQL: Schema validation already provided by UIMA typesystem
- Document oriented: CAS fits into document-oriented database

Available UIMA Components

- **MongoCollectionReader** reads CAS from a MongoDB collection. Optionally, a (filter) query can be specified;
- **RegexMongoCollectionReader**, similar to MongoCollectionReader but allows specifying a query with a regular expression on a specific field;
- **MongoWriter** persists new UIMA CASes into MongoDB documents;
- **MongoUpdateWriter** persists new annotations into an existing document;
- **MongoCollectionRemover** removes selected annotations in a MongoDB collection

Evaluation of MongoDB CAS Store

- Writes and reads performed on random sample of **500,000 PubMed abstracts** and annotated with all available Bluima NERs.
- Incremental annotation performed on a random sample of **5,000 PubMed abstracts** and incrementally annotated with the Stopwords annotator.
- Processing time and disk space measured on a **commodity laptop** (4 cores, 8GB RAM)
- Compared with XCAS, XMI, ZipXMI

Evaluation of MongoDB CAS Store

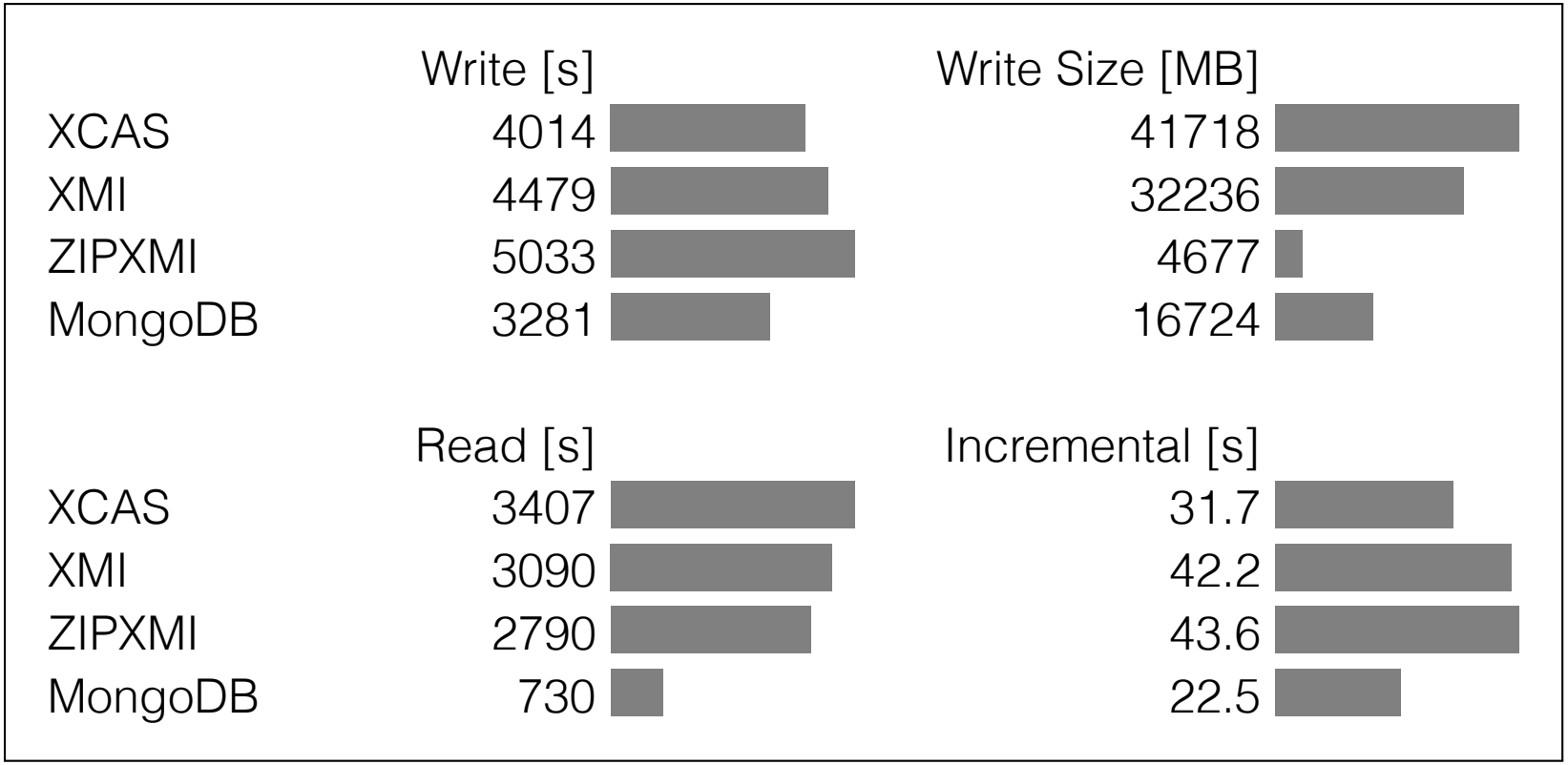


Fig. 1. Performance evaluation of MongoDB CAS Store against 3 other serialization formats.

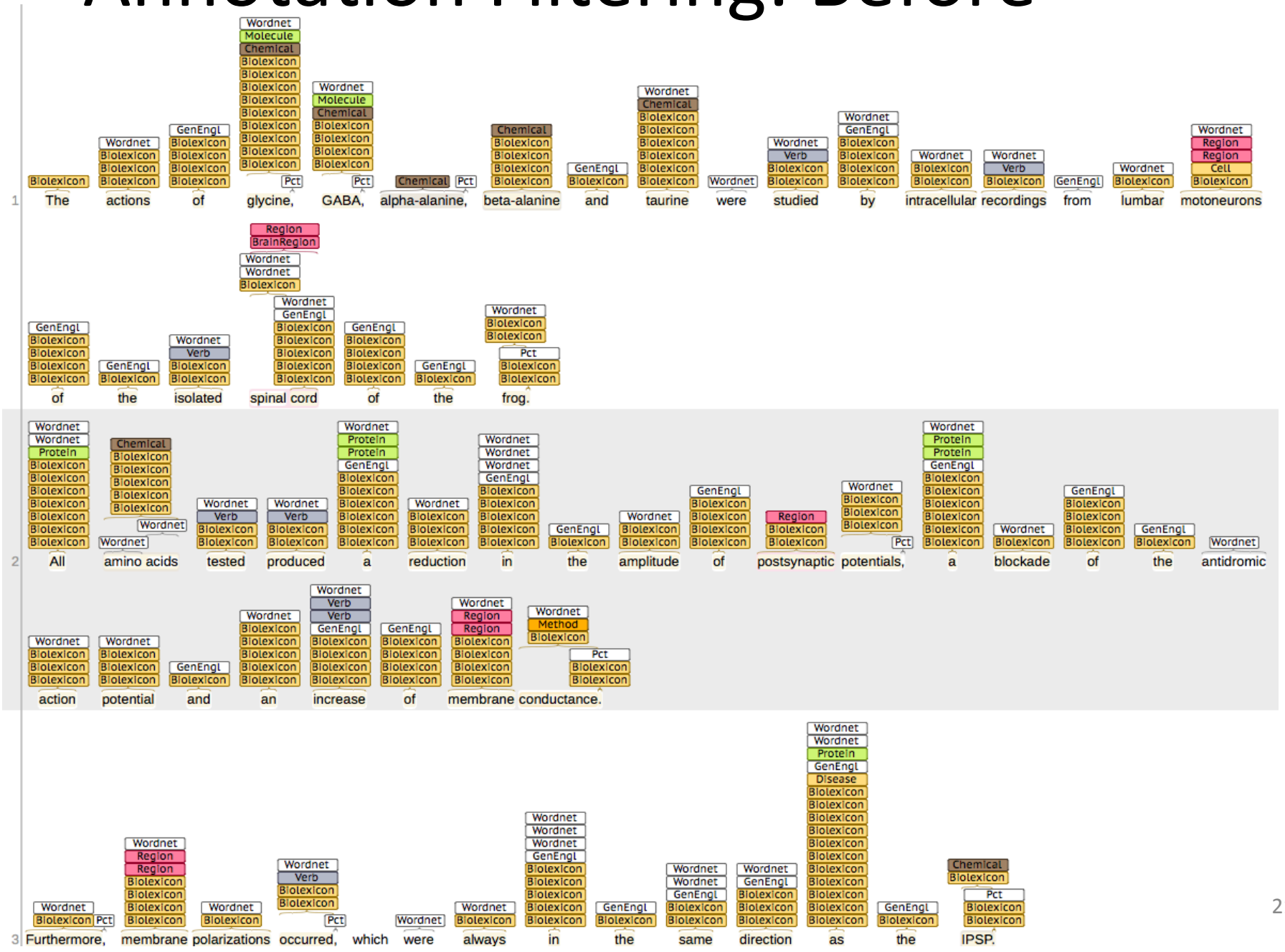
TODOs

- Complex types mapping
- Custom field mapping, or defaults
- Queries

Bluima Aussicht (Future work)

- Open source Bluima by end of year <-- PLEDGE!
- Annotation filtering
- Brain region NER (based on *French 2009*)
- Relation extraction for selected neuroscience entities
  - Brain region x brain region
  - Ion channel x subcellular location
  - Neuron properties
- LDA, hLDA
- Information extraction from PDF tables (supervised CRF)
- Scaling (full-text, Hadoop)

Annotation Filtering: Before



Annotation Filtering: After