Bluima: a UIMA-based NLP Toolkit for Neuroscience

Renaud Richardet, Jean-Cédric Chappelier, Martin Telefont

Blue Brain Project, EPFL, 1015, Lausanne, Switzerland
`renaud.richardet@epfl.ch`

Abstract. This paper describes *Bluima*, a natural language processing (NLP) pipeline focusing on the extraction of neuroscientific content and based on the UIMA framework. Bluima builds upon models from biomedical NLP (BioNLP) like specialized tokenizers and lemmatizers. It adds further models and tools specific to neuroscience (e.g. named entity recognizer for neuron or brain region mentions) and provides collection readers for neuroscientific corpora.
Two novel UIMA components are proposed: the first allows configuring and instantiating UIMA pipelines using a simple scripting language, enabling non-UIMA experts to design and run UIMA pipelines. The second component is a common analysis structure (CAS) store based on MongoDB, to perform incremental annotation of large document corpora.

Keywords: UIMA, natural language processing, NLP, neuroinformatics, NoSQL

1 Introduction

Bluima started as an effort to develop a high performance natural language processing (NLP) toolkit for neuroscience. The goal was to extract structured knowledge from biomedical literature (PubMed[1]), in order to help neuroscientists gather data to specify parameters for their models. In particular, focus was set on extracting entities that are specific to neuroscience (like brain regions and neurons) and that are not yet covered by existing text processing systems.

After careful evaluation of different NLP frameworks, the UIMA software system was selected for its open standards, its performance and stability, and its usage in several other biomedical NLP (bioNLP) projects; e.g. JulieLab [11], ClearTK [22], DKPRo [6], cTAKES [28], ccp-nlp, U-Compare [15], SciKnowMine [26], Argo [25]. Initial development went fast and several existing bioNLP models and UIMA components could rapidly be reused or integrated into UIMA without the need to modify its core system, as presented in Section 2.1.

Once the initial components were in place, an experimentation phase started where different pipelines were created, each with different components and parameters. Pipeline definition in verbose XML was greatly improved by the use

[1] http://www.ncbi.nlm.nih.gov/pubmed

of UIMAFit [21] (to define pipelines in compact Java code) but ended up being problematic, as it requires some Java knowledge and recompilation for each component or parameter change. To allow for a more agile prototyping, especially by non-specialist end users, a pipeline scripting language was created. It is described in Section 2.2.

Another concern was incremental annotation of large document corpus. For example, when running an initial pre-processing pipeline on several millions of documents, and then annotating them again at a later time. The initial strategy was to store the documents on disk, and overwrite them every time they would be incrementally annotated. Eventually, a CAS store module was developed to provide a stable and scalable strategy for incremental annotation, as described in Section 2.3. Finally, Section 3 presents two case studies illustrating the scripting language and evaluating the performance of the CAS store against existing serialization formats.

2 Bluima Components

Bluima contains several UIMA modules to read neuroscientific corpora, perform preprocessing, create simple configuration files to run pipelines, and persist documents on the disk.

2.1 UIMA Modules

Bluima's **typesystem** builds upon the typesystem from JulieLab [10], which was chosen for its strong biomedical orientation and its clean architecture. Bluima's typesystem adds neuroscientific annotations, like *CellType*, *BrainRegion*, etc.

Bluima includes several **collection readers** for selected neuroscience corpora, like PubMed XML dumps, PubMed Central NXML files, the BioNLP 2011 GENIA Event Extraction corpus [24], the Biocreative2 annotated corpus [16], the GENIA annotated corpus [14], and the WhiteText brain regions corpus [8]. A **PDF reader** was developed to provide robust and precise text extraction from scientific articles in PDF format. The PDF reader performs content correction and cleanup, like dehyphenation, removal of ligatures, glyph mapping correction, table detection, and removal of non-informative footers and headers.

For **pre-processing**, the OpenNLP-wrappers developed by JulieLab for sentence segmentation, word tokenization and part-of-speech tagging [31] were used and updated to UIMAFit. Lemmatization is performed by the domain-specific tool BioLemmatizer [19].Abbreviation recognition (the task of identifying abbreviations in text) is performed by BIOADI, a supervised machine learning model trained on the BIOADI corpus [17].

Bluima uses UIMA's ConceptMapper [29] to build **lexical-based NERs** based on several neuroscientific lexica and ontologies (Table 1). These lexica and ontologies were either developed in-house or were imported from existing sources. Bluima wraps several **machine learning-based NERs**, like OSCAR4 [13] (chemicals, reactions), Linnaeus [9] (species), BANNER [18] (genes and proteins), and Gimli [5] (proteins).

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.

2.2 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.

There are several approaches[2] to write and run UIMA pipelines (see Table 2). All Bluima components were initially written in Java with the UIMAFit library, that allows for compact code. To improve the design and experimentation with UIMA pipelines, and enable researchers without Java or UIMA knowledge to easily design and run such pipelines, a minimalistic scripting (domain-specific) language was developed, allowing UIMA pipelines to be configured with text files, in a human-readable format (Table 3). A *pipeline script* begins with the definition of a collection reader (starting with `cr:`), followed by several annotation engines (starting with `ae:`)[3]. Parameter specification starts with a space, followed by the

[2] Other interesting solutions exist (e.g. IBM LanguageWare, Argo), but are not open source.

[3] If not package namespace is specified, Bluima loads Readers and Annotator classes from the default namespace.

parameter name, a column and its value. The scripting language also supports embedding of inline Python and Java code, reuse of a portion of a pipeline with `include` statements, and variable substitution similar to shell scripts. Extensive documentation (in particular snippets of scripts) is automatically generated for all components, using the JavaDoc and the UIMAFit annotations.

2.3 CAS Store

A CAS store was developed to persist annotated documents, resume their processing and add new annotations to them. This CAS store was motivated by the common use case of repetitively and incrementally processing the same documents with different UIMA pipelines, where some pipeline steps are duplicated among the runs. For example, when performing resource-intensive operations (like extracting the text from full-text PDF articles, or performing syntactic parsing), one might want to perform these preliminary operation once, store these results, and subsequently perform different experiments with different UIMA modules and parameters. The CAS store thus allows to perform the preprocessing only once, to then persist the annotated documents, and to perform the various experiments in parallel.

MongoDB[4] was selected as the datastore backend. MongoDB is a scalable, high-performance, open-source, schema-free (NoSQL), document-oriented database. No schema is required on the database side, since the UIMA typesystem acts as a schema, and data is validated on-the-fly by the module. Every CAS is stored as a MongoDB document, along with its annotations. UIMA annotations and their features are explicitly mapped to MongoDB fields, using a simple and declarative language. For example, a `Protein` annotation is mapped to a `prot` field in MongoDB. The mappings are used when persisting and loading from the database. As of this writing, annotations are declared in Java source files. In future versions, we plan to store mappings directly in MongoDB to improve flexibility. Persistence of complex typesystem has not been implemented yet, but could be easily added in the future.

Currently, the following UIMA components are available for the CAS store:

- *MongoCollectionReader* reads CAS from a MongoDB collection. Optionally, a (filter) query can be specified;
- *RegexMongoCollectionReader* is 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.

With the above components, it is possible within a single pipeline to read an existing collection of annotated documents, perform some further processing, add more annotations, and store theses annotations back into the same MongoDB documents.

[4] http://www.mongodb.org/

3 Case Studies and Evaluation

A first experiment to illustrate the scripting language was conducted on a large dataset of full-text biomedical articles. A second simulated experiment evaluates the performance of the MongoDB CAS store against existing serialization formats.

3.1 Scripting and Scale-Out

```
# 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.

Bluima was used to extract brain region mention co-occurrences from scientific articles in PDF. The pipeline script (Table 3) was created and tested on a development laptop. Scale-out was performed on a 12-node (144-core) cluster managed by SLURM (Simple Linux Utility for Resource Management). The 383,795 PDFs were partitioned in 767 jobs. Each job was instantiated with the same pipeline script, using different input and output parameters. The processing completed in 809 minutes ($\simeq$ 8 PDF/s).

3.2 MongoDB CAS Store

The MongoDB CAS store (MCS) has been evaluated against 3 other available serialization formats (XCAS, XMI and ZIPXMI). For each, 3 settings were evaluated: writes (CASes are persisted to disk), reads (CASes are loaded from their persisted states), and incremental (CASes are first read from their persisted

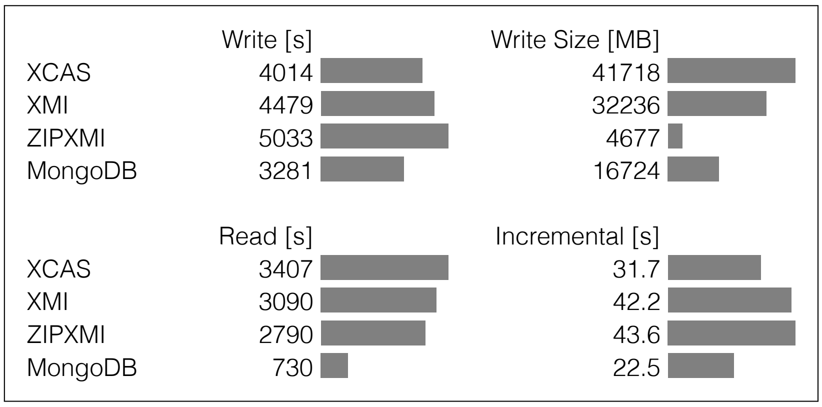

Fig. 1. Performance evaluation of MongoDB CAS Store against 3 other serialization formats.

states, then further processed, and finally persisted again to disk). Writes and reads were performed on a random sample of 500,000 PubMed abstracts and annotated with all available Bluima NERs. Incremental annotation was performed on a random sample of 5,000 PubMed abstracts and incrementally annotated with the `Stopwords` annotator. Processing time and disk space was measured on a commodity laptop (4 cores, 8GB RAM).

In terms of speed, the MCS significantly outperforms the other formats, especially for reads (Figure 1). The MCS disk size is significantly smaller than XCAS and XMI formats, but almost 4 times larger than the compressed ZIPXMI format. The incremental annotation is significantly faster with MongoDB, and does not require duplicating or overwriting files, like with the other serialization formats. The MCS could be scaled up in a cluster setup, or using solid states drives (SSDs). Writes could probably be improved by turning MongoDB's "safe mode" option off. Furthermore, by adding indexes, the MCS can act as a searchable annotation database.

4 Conclusions and Future Work

In the process of developing Bluima, a toolkit for neuroscientific NLP, we integrated and wrapped several specialized resources to process neuroscientific articles. We also created two UIMA modules (scripting language and CAS store). These additions proved to be very effective in practice and allowed us to leverage UIMA, an enterprise-grade framework, while at the same time allowing an agile development and deployment of NLP pipelines.

In the future, we will open-source Bluima and add more models for NER and relationship extraction. We also plan to ease the deployment of Bluima (and its scripting language) on a Hadoop cluster.

References

1. Bairoch, A., Apweiler, R., Wu, C.H., Barker, W.C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M.: The universal protein resource (UniProt). Nucleic acids research 33(suppl 1), D154–D159 (2005)
2. Bard, J., Rhee, S.Y., Ashburner, M.: An ontology for cell types. Genome Biology 6(2) (2005)
3. Bowden, D., Dubach, M.: NeuroNames 2002. Neuroinformatics 1(1), 43–59 (2003)
4. Bug, W.J., Ascoli, G.A., Grethe, J.S., Gupta, A., Fennema-Notestine, C., Laird, A.R., Larson, S.D., Rubin, D., Shepherd, G.M., Turner, J.A.: The NIFSTD and BIRNLex vocabularies: building comprehensive ontologies for neuroscience. Neuroinformatics 6(3), 175–194 (2008)
5. Campos, D., Matos, S., Oliveira, J.L.: Gimli: open source and high-performance biomedical name recognition. BMC Bioinformatics 14(1), 54 (Feb 2013)
6. De Castilho, R.E., Gurevych, I.: DKPro-UGD: a flexible data-cleansing approach to processing user-generated discourse. In: Onlineproceedings of the First French-speaking meeting around the framework Apache UIMA, LINA CNRS UMR (2009)
7. Fellbaum, C.: WordNet. Theory and Applications of Ontology: Computer Applications p. 231–243 (2010)
8. French, L., Lane, S., Xu, L., Pavlidis, P.: Automated recognition of brain region mentions in neuroscience literature. Front Neuroinformatics 3 (Sep 2009)
9. Gerner, M., Nenadic, G., Bergman, C.: Linnaeus: A species name identification system for biomedical literature. BMC Bioinformatics 11(1), 85 (2010)
10. Hahn, U., Buyko, E., Tomanek, K., Piao, S., Mcnaught, J., Tsuruoka, Y., Ananiadou, S.: An Annotation Type System for a Data-Driven NLP Pipeline (2007)
11. Hahn, U., Buyko, E., Landefeld, R., Mühlhausen, M., Poprat, M., Tomanek, K., Wermter, J.: An overview of JCoRe, the JULIE lab UIMA component repository. In: Proceedings of the LREC. vol. 8, p. 1–7 (2008)
12. Imam, F.T., Larson, S.D., Grethe, J.S., Gupta, A., Bandrowski, A., Martone, M.E.: NIFSTD and NeuroLex: a comprehensive neuroscience ontology development based on multiple biomedical ontologies and community involvement (2011)
13. Jessop, D., et al.: OSCAR4: a flexible architecture for chemical text-mining. Journal of Cheminformatics 3(1), 41 (Oct 2011)
14. Kim, J.D., Ohta, T., Tateisi, Y., Tsujii, J.: GENIA corpus–a semantically annotated corpus for bio-textmining. Bioinformatics 19, i180–i182 (Jul 2003)
15. Kontonatsios, G., Korkontzelos, I., Kolluru, B., Thompson, P., Ananiadou, S.: Deploying and sharing u-compare workflows as web services. J. Biomedical Semantics 4, 7 (2013)
16. Krallinger, M., Morgan, A., Smith, L., Leitner, F., Tanabe, L., Wilbur, J., Hirschman, L., Valencia, A.: Evaluation of text-mining systems for biology: overview of the second BioCreative community challenge. Genome Biology 9(Suppl 2), S1 (2008)
17. Kuo, C.J., et al.: BioAdi: a machine learning approach to identifying abbreviations and definitions in biological literature. BMC Bioinformatics 10(Suppl 15), S7 (Dec 2009)
18. Leaman, R., Gonzalez, G., et al.: BANNER: an executable survey of advances in biomedical named entity recognition. In: Pacific Symposium on Biocomputing. vol. 13, p. 652–663 (2008)
19. Liu, H., et al.: BioLemmatizer: a lemmatization tool for morphological processing of biomedical text. Journal of Biomedical Semantics 3(1), 3 (Apr 2012)

20. Natale, D.A., Arighi, C.N., Barker, W.C., Blake, J.A., Bult, C.J., Caudy, M., Drabkin, H.J., D'Eustachio, P., Evsikov, A.V., Huang, H., Nchoutmboube, J., Roberts, N.V., Smith, B., Zhang, J., Wu, C.H.: The protein ontology: a structured representation of protein forms and complexes. Nucleic Acids Res. 39(Database issue), D539–545 (Jan 2011)
21. Ogren, P.V., Bethard, S.J.: Building test suites for UIMA components. NAACL HLT 2009 p. 1 (2009)
22. Ogren, P.V., Wetzler, P.G., Bethard, S.J.: ClearTK: a UIMA toolkit for statistical natural language processing. Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP p. 32 (2008)
23. Osborne, J., Flatow, J., Holko, M., Lin, S.M., Kibbe, W.A., Zhu, L.J., Danila, M.I., Feng, G., Chisholm, R.L.: Annotating the human genome with disease ontology. BMC Genomics 10(Suppl 1), S6 (Jul 2009)
24. Pyysalo, S., Ohta, T., Rak, R., Sullivan, D., Mao, C., Wang, C., Sobral, B., Tsujii, J., Ananiadou, S.: Overview of the ID, EPI and REL tasks of BioNLP shared task 2011. BMC Bioinformatics 13(Suppl 11), S2 (Jun 2012)
25. Rak, R., Rowley, A., Black, W., Ananiadou, S.: Argo: an integrative, interactive, text mining-based workbench supporting curation. Database: the journal of biological databases and curation 2012 (2012)
26. Ramakrishnan, C., Baumgartner Jr, W.A., Blake, J.A., Burns, G.A., Cohen, K.B., Drabkin, H., Eppig, J., Hovy, E., Hsu, C.N., Hunter, L.E.: Building the scientific knowledge mine (SciKnowMine1): a community-driven framework for text mining tools in direct service to biocuration. malta. Language Resources and Evaluation (2010)
27. Ranjan, R., Khazen, G., Gambazzi, L., Ramaswamy, S., Hill, S.L., Schürmann, F., Markram, H.: Channelpedia: an integrative and interactive database for ion channels. Frontiers in neuroinformatics 5 (2011)
28. Savova, G.K., Masanz, J.J., Ogren, P.V., Zheng, J., Sohn, S., Kipper-Schuler, K.C., Chute, C.G.: 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)
29. Tanenblatt, M.A., Coden, A., Sominsky, I.L.: The ConceptMapper approach to named entity recognition. In: LREC (2010)
30. Thompson, P., et al.: The BioLexicon: a large-scale terminological resource for biomedical text mining. BMC Bioinformatics 12(1), 397 (2011)
31. Tomanek, K., Wermter, J., Hahn, U.: A reappraisal of sentence and token splitting for life sciences documents. Studies in health technology and informatics 129(Pt 1), 524–528 (2006)