TOWARDS DATA AND GOAL ORIENTED ANALYSIS: TOOL INTER-OPERABILITY AND COMBINATORIAL COMPARISON

Yoshinobu Kano[1] **Ngan Nguyen**[1] **Rune Sætre**[1] **Kazuhiro Yoshida**[1]
Keiichiro Fukamachi[1] **Yusuke Miyao**[1] **Yoshimasa Tsuruoka**[3]
Sophia Ananiadou[2,3] **Jun'ichi Tsujii**[1,2,3]

[1]Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 Tokyo

[2]School of Computer Science, University of Manchester
PO Box 88, Sackville St, MANCHESTER M60 1QD, UK

[3]NaCTeM (National Centre for Text Mining), Manchester Interdisciplinary Biocentre,
University of Manchester, 131 Princess St, MANCHESTER M1 7DN, UK

{kano,nltngan,satre,kyoshida,keif,yusuke,tsujii}
@is.s.u-tokyo.ac.jp
{yoshimasa.tsuruoka,sophia.ananiadou}@manchester.ac.uk

Abstract

Recently, NLP researches have advanced using F-scores, precisions, and recalls with gold standard data as evaluation measures. However, such evaluations cannot capture the different behaviors of varying NLP tools or the different behaviors of a NLP tool that depends on the data and domain in which it works. Because an increasing number of tools are available nowadays, it has become increasingly important to grasp these behavioral differences, in order to select a suitable set of tools, which forms a complex workflow for a specific purpose. In order to observe such differences, we need to integrate available combinations of tools into a workflow and to compare the combinatorial results. Although generic frameworks like UIMA (Unstructured Information Management Architecture) provide interoperability to solve this problem, the solution they provide is only partial. In order for truly interoperable toolkits to become a reality, we also need sharable and comparable type systems with an automatic combinatorial comparison generator, which would allow systematic comparisons of available tools. In this paper, we describe such an environment, which we developed based on UIMA, and we show its feasibility through an example of a protein-protein interaction (PPI) extraction system.

1 Introduction

Recently, an increasing number of TM/NLP tools such as part-of-speech (POS) taggers (Tsuruoka et al., 2005), named entity recognizers (NERs) (Settles, 2005) syntactic parsers (Hara et al., 2005) and relation or event extractors (ERs) have been developed. Nevertheless, it is still very difficult to integrate independently developed tools into an aggregated application that achieves a specific task. The difficulties are caused not only by differences in programming platforms and different input/output data formats, but also by the lack of higher level interoperability among modules developed by different groups.

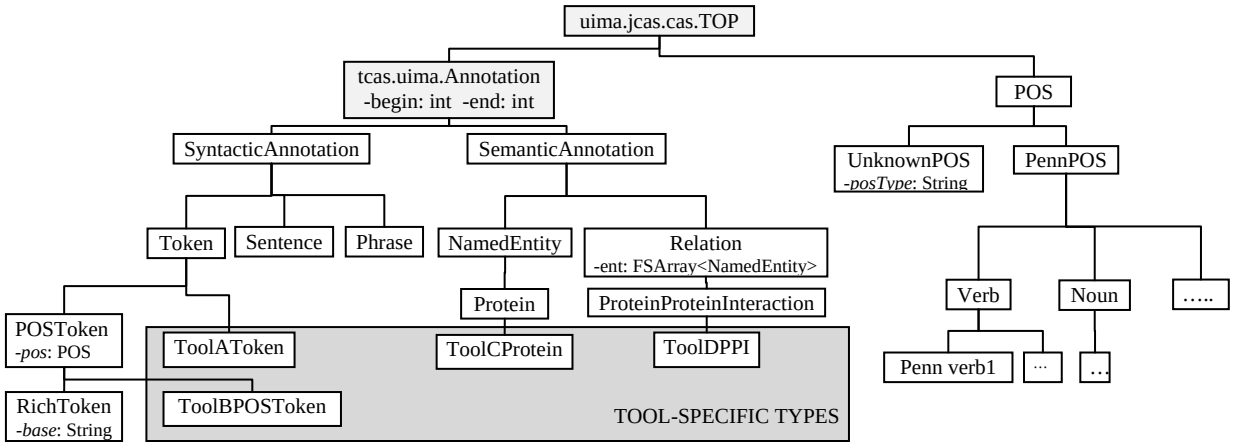

Figure 1. Part of our type system

UIMA, Unstructured Information Management Architecture (Lally and Ferrucci, 2004), which was originally developed by IBM and has recently become an open project in OASIS and Apache, provides a promising framework for tool integration. Although it has a set of useful functionalities, UIMA only provides a generic framework, thus it requires a user community to develop their own platforms with a set of actual software modules. A few attempts have already been made to establish platforms, e.g. the CMU UIMA component repository [1], GATE (Cunningham et al., 2002) with its UIMA interoperability layer, etc.

However, simply wrapping existing modules to be UIMA compliant does not offer a complete solution. Most of TM/NLP tasks are composite in nature, and can only be solved by combining several modules. Users need to test a large number of combinations of tools in order to pick the most suitable combination for their specific task.

Although *types* and *type systems* are the only way to represent meanings in the UIMA framework, UIMA does not provide any specific *types*, except for a few purely primitive *types*. In this paper, we propose a way to design sharable *type systems*. A sharable *type system* designed in this way can provide the interoperability between independently developed tools with fewer losses in information, thus allowing for the combinations of tools and comparisons on these combinations.

We show how our automatic comparison generator works based on a *type system* designed in that way. Taking the extraction of protein-protein interaction (PPI) as a typical example of a composite task, we illustrate how our platform helps users to observe the differences between tools and to construct a system for their own needs.

2 Motivation and Background

2.1 Goal and Data Oriented Evaluation, Module Selection and Inter-operability

There are standard evaluation metrics for NLP modules such as precision, recall and F-value. For basic tasks like sentence splitting, POS tagging, and named-entity recognition, these metrics can be estimated using existing gold-standard test sets.

Conversely, accuracy measurements based on the standard test sets are sometimes deceptive, since its accuracy may change significantly in practice, depending on the types of text and the actual tasks at hand. Because these accuracy metrics do not take into account the importance of the different types of errors to any particular application, the practical utility of two systems with seemingly similar levels of accuracy may in fact differ significantly. To users and developers alike, a detailed examination of how systems perform (on the text they would like to process) is often more important than standard metrics and test sets. Naturally, far greater weight is placed in measuring the end-to-end performance of a composite system than in measuring the performance of the individual components.

In reality, because the selection of modules usually affects the performance of the entire system, it is crucial to carefully select modules that are appropriate for a given task. This is the main reason for having a collection of interoperable

[1] http://uima.lti.cs.cmu.edu/

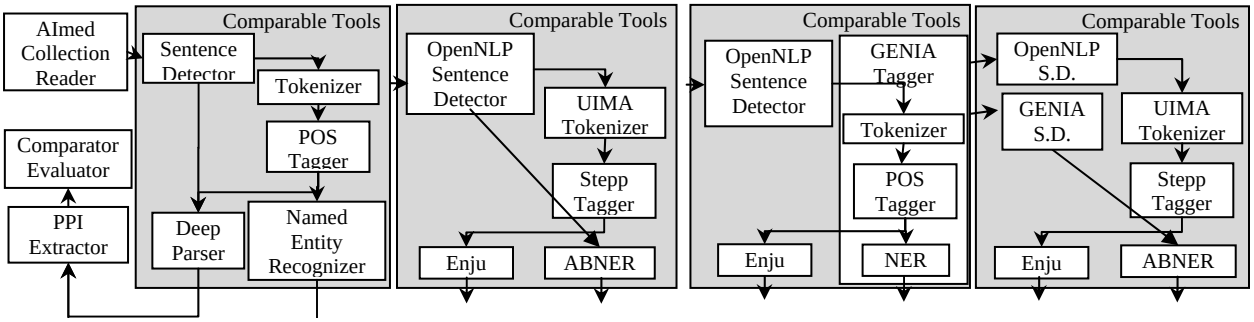

Figure 2. PPI system workflow (conceptual)

Figure 3. Basic example pattern

Figure 4. Complex tool example

Figure 5. Branch flow pattern

modules. We need to show how the ultimate performance will be affected by the selection of different modules and show the best combination of modules in terms of the performance of the whole aggregated system for the task at hand.

Since the number of possible combinations of component modules is typically large, the system has to be able to enumerate and execute them semi-automatically. This requires a higher level of interoperability of individual modules than just wrapping them for UIMA.

2.2 UIMA

2.2.1 CAS and Type System

The UIMA framework uses the "stand-off annotation" style (Ferrucci et al., 2006). The raw text in a document is kept unchanged during the analysis process, and when the processing of the text is performed, the result is added as new stand-off annotations with references to their positions in the raw text. A Common Analysis Structure (CAS) maintains a set of these annotations, which in itself are objects. The annotation objects in a CAS belong to *types* that are defined separately in a hierarchical *type system*. The features of an *annotation*[2] object have values that are typed as well.

2.2.2 Component and Capability

Each UIMA Component has the *capability* property which describes what types of objects the component may take as the input and what types of objects it produces as the output. For example, a named entity recognizer detects named entities in the text and outputs annotation objects of the type `NamedEntity`.

It is possible to deploy any UIMA component as a SOAP web service, so that we can combine a remote component on a web service with the local component freely inside a UIMA-based system.

3 Integration Platform and Comparators

3.1 Sharable and Comparable Type System

Although UIMA provides a set of useful functionalities for an integration platform of TM/NLP tools, users still have to develop the actual platform by using these functionalities effectively. There are several decisions for the designer to make an integration platform.

Determining how to use *types* in UIMA is a crucial decision. Our decision is to keep different *type systems* by individual groups as they are, if necessary; we require that individual *type systems* have to be related through a sharable *type system*, which our platform defines. Such a shared *type system* can bridge modules with different *type systems*, though the bridging module may lose some information during the translation process.

Whether such a sharable *type system* can be defined or not is dependent on the nature of each problem. For example, a sharable *type system* for POS tags in English can be defined rather easily, since most of POS-related modules (such as POS taggers, shallow parsers, etc.) more or less follow the well established types defined by the Penn Treebank (Marcus et al., 1993) tag set.

Figure 1 shows a part of our sharable *type system*. We deliberately define a highly organized type hierarchy as described above.

Secondly we should consider that the *type system* may be used to compare a similar sort of tools. *Types* should be defined in a distinct and

[2] In the UIMA framework, `Annotation` is a base *type* which has *begin* and *end* offset values. In this paper we call any objects (any subtype of `TOP`) as *annotations*.

hierarchical manner. For example, both tokenizers and POS taggers output an object of *type* `Token`, but their roles are different when we assume a cascaded pipeline. We defined `Token` as a supertvpe, `POSToken` as subtypes of `Token`. Each tool should have an individual *type* to make clear which tool generated which instance, because each tool may have a slightly different definition. This is important because the *capabilities* are represented by these *types*, and the *capabilities* are the only attributes which are machine readable.

3.2 General Combinatorial Comparison Generator

Even if the *type system* is defined in the previously described way, there are still some issues to consider when comparing tools. We illustrate these issues using the PPI *workflow* that we utilized in our experiments.

Figure 2 conceptually shows the *workflow* of our whole PPI system. If we can prepare two or more components for some type of the components in the *workflow* (e.g. two sentence detectors and three POS taggers), then we can make combinations of these tools to form a multiplied number of *workflow* patterns (2x3 = 6 patterns). See Table 1 for the details of UIMA components used in our experiments.

We made a pattern expansion mechanism which generates possible *workflow* patterns automatically from a user-defined *comparable workflow*. A *comparable workflow* is a special *workflow* that explicitly specifies which set of components should be compared. Then, users just need to group comparable components (e.g. ABNER[3] and MedT-NER as a comparable NER group) without making any modifications to the original UIMA components. This aggregation of comparable components is controlled by our *custom workflow controller*.

In some cases, a single tool can play two or more roles (e.g. the GENIA Tagger performs tokenization, POS tagging, and NER; see Figure 4). It may be possible to decompose the original tool into single roles, but in most cases it is difficult and unnatural to decompose such a complex tool. We designed our comparator to detect possible input combinations automatically by the *types* of previously generated *annotations*, and the input *capability* of each posterior component. As described in the previous section, the component should have appropriate *capabilities* with proper *types* in order to permit this detection.

When a component requires two or more input *types* (e.g. our PPI extractor requires outputs of a deep parser and a protein NER system), there could be different components used in the prior flow (e.g. OpenNLP and GENIA sentence detectors in Figure 5). Our comparator also calculates such cases automatically.

	O	U	G	A
G	0	0	-	85
U	86	-	0	7
A	6	6	60	-
O	-	81	0	7

Table 2. `Sentence` comparisons (%).

	OO	UO	GO
UU	89/75	89/75	88/70
GU	89/75	89/75	88/70
GG	92/95	91/95	97/95
OG	100/100	99/99	100/94

Table 3. Part of `Token` comparisons, precision/recall (%).

	OOO	UOS	GOO
UUO	87/74	81/68	85/68
GUG	74/65	73/65	78/65
GGO	92/95	81/84	97/95
OGO	100/100	89/88	100/94

Table 4. Part of `POSToken` comparisons, precision/recall (%)

4 Experiments and Results

We have performed experiments using our PPI extraction system as an example (Kano et al., 2008). It is similar to our BioCreative PPI system (Sætre et al., 2006) but differs in that we have deconstructed the original system into seven different components (Figure 2).

As summarized in Table 1, we have several comparable components and the AImed corpus as the gold standard data. In this case, possible combination *workflow* patterns are `POSToken` for 36, `PPI` for 589, etc.

Table 2, 3, 4 and Figure 6 show a part of the comparison result screenshots between these patterns on 20 articles from the AImed corpus. In the tables, abbreviations like "OOG" stands for a workflow of O(`Sentence`) -> O(`Token`) -

[3] In the example figures, ABNER requires `Sentence` to make the explanation clearer, though ABNER does not require it in actual usage.

G(POSToken), where O stands for OpenNLP, G stands for Genia, U stands for UIMA, etc.

When neither of the compared results include the gold standard data (AImed in this case), the comparison results show a *similarity* of the tools for this specific task and data, rather than an evaluation. Even if we lack an annotated corpus, it is possible to run the tools and compare the results in order to understand the characteristics of the tools depending on the corpus and the tool combinations.

Although the comparison on Sentences shows low scores of *similarities*, Tokens are almost the same; it means that input sentence boundaries do not affect tokenizations so much. POSToken *similarities* drop approximately 0-10% from the *similarities* in Token; the differences in Token are mainly apostrophes and punctuations; POSTokens are different because each POS tagger uses a slightly different set of tags: normal Penn tagset for Stepp tagger, BioPenn tagset (includes new tags for hyphenation) for GENIA tagger, and an original apostrophe tag for OpenNLP tagger.

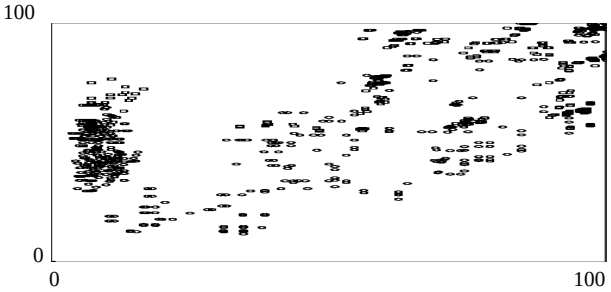

Figure 6. NER (Protein) comparison distribution of precisions (x-axis, %) and recalls (y-axis, %).

5 Conclusion and Future Work

NLP tasks typically consist of many components, and it is necessary to show which set of tools are most suitable for each specific task and data. Although UIMA provides a general framework with much functionality for interoperability, we still need to build an environment that enables the combinations and comparisons of tools for a specific task.

The *type system* design, which the UIMA framework does not provide, is one of the most critical issues on interoperability. We have thus proposed a way to design a sharable and comparable *type system*. Such a *type system* allows for the automatic combinations of any UIMA compliant components and for the comparisons of these combinations, when the components have proper *capabilities* within the *type system*. We are

Sentence	Token	POSToken	RichToken	Protein	Phrase	PPI
GENIA Tagger: Trained on the WSJ, GENIA and PennBioIE corpora (POS). Uses Maximum Entropy (Berger et al., 1996) classification, trained on JNLPBA (Kim et al., 2004) (NER). Trained on GENIA corpus (Sentence Splitter).						
Enju: HPSG parser with predicate argument structures as well as phrase structures. Although trained with Penn Treebank, it can compute accurate analyses of biomedical texts owing to its method for domain adaptation (Hara et al., 2005).						
STePP Tagger: Based on probabilistic models, tuned to biomedical text trained by WSJ, GENIA (Kim et al., 2003) and PennBioIE corpora.						
MedT-NER: Statistical recognizer trained on the JNLPBA data.						
ABNER: From the University of Wisconsin (Settles, 2005), wrapped by the Center for Computational Pharmacology at the University of Colorado.						
Akane++: A new version of the AKANE system (Yakushiji, 2006), trained with SVMlight-TK (Joachims, 1999; Bunescu and Mooney, 2006; Moschitti, 2006) and the AImed Corpus.						
UIMA Examples: Provided in the Apache UIMA example. Sentence Splitter and Tokenizer.						
OpenNLP Tools: Part of the OpenNLP project (http://opennlp.sourceforge.net/), from Apache UIMA examples.						
AImed Corpus: 225 Medline abstracts with proteins and PPIs annotated (Bunescu and Mooney, 2006).						

Legend: Input type(s) required for that tool Input type(s) required optionally Output type(s)

Table 1. List of UIMA Components used in our experiment.

preparing to make a portion of the components and services described in this paper publicly available (http://www-tsujii.is.s.u-tokyo.ac.jp/uima/).

The final system shows which combination of components has the best score, and also generates comparative results. This helps users to grasp the characteristics and differences among tools, which cannot be easily observed by the widely used F-score evaluations only.

Future directions for this work includes combining the output of several modules of the same kind (such as NERs) to obtain better results, collecting other tools developed by other groups using the sharable *type system*, making machine learning tools UIMA compliant, and making grid computing available with UIMA workflows to increase the entire performance without modifying the original UIMA components.

Acknowledgments

We wish to thank Dr. Lawrence Hunter's text mining group at the Center for Computational Pharmacology for discussing with us and making their tools available for this research. This work was partially supported by NaCTeM (the UK National Centre for Text Mining), Grant-in-Aid for Specially Promoted Research (MEXT, Japan) and Genome Network Project (MEXT, Japan). NaCTeM is jointly funded by JISC/BBSRC/EPSRC.

References

Berger, Adam L., Vincent J. Della Pietra, and Stephen A. Della Pietra. "A maximum entropy approach to natural language processing." *Comput. Linguist.* (MIT Press) 22, no. 1 (1996): 39-71.

Bunescu, Razvan, and Raymond Mooney. "Subsequence Kernels for Relation Extraction." Edited by Weiss Y., Scholkopf B. and Platt J., 171-178. Cambridge, MA: MIT Press, (2006).

Cunningham, H., D. Maynard, K. Bontcheva, and V. Tablan. "GATE: A framework and graphical development environment for robust NLP tools and applications." *Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics.* (2002).

Ferrucci, David, Adam Lally, Daniel Gruhl, and Edward Epstein. "Towards an Interoperability Standard for Text and Multi-Modal Analytics." IBM Research Report, RC24122. (2006).

Hara, Tadayoshi, Yusuke Miyao, and Jun'ichi Tsujii. "Adapting a probabilistic disambiguation model of an HPSG parser to a new domain." Edited by Dale Robert, Wong Kam-Fai, Su Jian and Yee Oi. *Natural Language Processing IJCNLP 2005.* Jeju Island, Korea: Springer-Verlag, (2005). 199-210.

Joachims, Thorsten. "Making large-scale support vector machine learning practical." MIT Press, (1999): 169-184.

Kano, Yoshinobu, et al. "Filling the gaps between tools and user: a tool comparator, using protein-protein interaction as an example." *Proceedings of The Pacific Symposium on Biocomputing (PSB).* Hawaii, USA, To appear, (2008).

Kim, Jin-Dong, Tomoko Ohta, Yoshimasa Tsuruoka, and Yuka Tateisi. "Introduction to the Bio-Entity Recognition Task at JNLPBA." *Proceedings of the International Workshop on Natural Language Processing.* Geneva, Switzerland, (2004). 70-75.

Kim, Jin-Dong, Tomoko Ohta, Yuka Teteisi, and Jun'ichi Tsujii. "GENIA corpus - a semantically annotated corpus for bio-textmining." *Bioinformatics* (Oxford University Press) 19, no. suppl. 1 (2003): i180-i182.

Lally, Adam, and David Ferrucci. "Building an Example Application with the Unstructured Information Management Architecture." *IBM Systems Journal* 43, no. 3 (2004): 455-475.

Marcus, Mitchell P., Beatrice Santorini, and Mary Ann Marcinkiewicz. "Building a Large Annotated Corpus of English: The Penn Treebank." *Computational Linguistics* 19, no. 2 (1993): 313-330.

Moschitti, Alessandro. "Making Tree Kernels Practical for Natural Language Learning." *EACL.* (2006).

Sætre, Rune, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yuichiroh Matsubayashi, and Tomoko Ohta. "AKANE System: Protein-Protein Interaction Pairs in BioCreAtIvE2 Challenge." *Proceedings of the Second BioCreative Challenge Evaluation Workshop.* (2007).

Settles, B. "ABNER: an open source tool for automatically tagging genes, proteins, and other entity names in text." *Bioinformatics* (Oxford University Press) 21, no. 14 (2005): 3191-3192.

Tsuruoka, Yoshimasa, Yuka Tateishi, Jin-Dong Kim, and Tomoko Ohta. "Developing a Robust Part-of-Speech Tagger for Biomedical Text." *Advances in Informatics - 10th Panhellenic Conference on Informatics.* Volos, Greece, (2005). 382-392.

Yakushiji, Akane. "Relation Information Extraction Using Deep Syntactic Analysis." PhD Thesis, University of Tokyo, (2006).