Integrated NLP Evaluation System for Pluggable Evaluation Metrics with Extensive Interoperable Toolkit

Yoshinobu Kano[1] **Luke McCrohon**[1] **Sophia Ananiadou**[2] **Jun'ichi Tsujii**[1,2]

[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 and National Centre for Text Mining, 131 Princess St, M1 7DN, UK

[kano,tsujii]@is.s.u-tokyo.ac.jp
luke.mccrohon@gmail.com
sophia.ananiadou@manchester.ac.uk

Abstract

To understand the key characteristics of NLP tools, evaluation and comparison against different tools is important. And as NLP applications tend to consist of multiple semi-independent sub-components, it is not always enough to just evaluate complete systems, a fine grained evaluation of underlying components is also often worthwhile. Standardization of NLP components and resources is not only significant for reusability, but also in that it allows the comparison of individual components in terms of reliability and robustness in a wider range of target domains. But as many evaluation metrics exist in even a single domain, any system seeking to aid inter-domain evaluation needs not just predefined metrics, but must also support pluggable user-defined metrics. Such a system would of course need to be based on an open standard to allow a large number of components to be compared, and would ideally include visualization of the differences between components. We have developed a pluggable evaluation system based on the UIMA framework, which provides visualization useful in error analysis. It is a single integrated system which includes a large ready-to-use, fully interoperable library of NLP tools.

1 Introduction

When building NLP applications, the same subtasks tend to appear frequently while constructing different systems. Due to this, the reusability of tools designed for such subtasks is a common design consideration; fine grained interoperability between sub components, not just between complete systems.

In addition to the benefits of reusability, interoperability is also important in evaluation of components. Evaluations are normally done by comparing two sets of data, a gold standard data and test data showing the components performance. Naturally this comparison requires the two data sets to be in the same data format with the same semantics. Comparing of "Apples to Apples" provides another reason why standardization of NLP tools is beneficial. Another advantage of standardization is that the number of gold standard data sets that can be compared against is also increased, allowing tools to be tested in a wider range of domains.

The ideal is that all components are standardized to conform to an open, widely used interoperability framework. One possible such framework is UIMA; Unstructured Information Management Architecture (Ferrucci et al., 2004), which is an open project of OASIS and Apache. We have been developing U-Compare (Kano et al., 2009)[1], an integrated testing an evaluation platform based on this framework.

[1] Features described in this paper are integrated as U-Compare system, publicly available from: http://u-compare.org/

Proceedings of the NAACL HLT Workshop on Software Engineering, Testing, and Quality Assurance for Natural Language Processing, pages 22–30, Boulder, Colorado, June 2009.

Although U-Compare already provided a wide range of tools and NLP resources, its inbuilt evaluation mechanisms were hard coded into the system and were not customizable by end users. Furthermore the evaluation metrics used were based only on simple strict matchings which severely limited its domains of application. We have extended the evaluation mechanism to allow users to define their own metrics which can be integrated into the range of existing evaluation tools.

The U-Compare library of interoperable tools has also been extended; especially with regard to resources related to biomedical named entity extraction. U-Compare is currently providing the world largest library of type system compatible UIMA components.

In section 2 of this paper we first look at the underlying technologies, UIMA and U-Compare. Then we describe the new pluggable evaluation mechanism in section 3 and our interoperable toolkit with our type system in section 4 and 5.

2 Background

2.1 UIMA

UIMA is an open framework specified by OASIS[2]. Apache UIMA provides a reference implementation as an open source project, with both a pure java API and a C++ development kit . UIMA itself is intended to be purely a framework, i.e. it does not intend to provide specific tools or type system definitions. Users should develop such resources themselves. In the following subsections, we briefly describe the basic concepts of UIMA, and define keywords used to explain our system in later sections.

2.1.1 CAS and Type System

The UIMA framework uses the "stand-off annotation" style (Ferrucci et al., 2006). The underling raw text of a document is generally kept unchanged during analysis, and the results of processing the text are added as new stand-off annotations with references to their positions in the raw text. A *Common Analysis Structure* (*CAS*) holds a set of such annotations. Each of which is of a given *type* as defined in a specified hierarchical *type system*. Annotation[3] types may define features, which are themselves typed. Apache UIMA provides definitions of a range of built in primitive types, but a more complete type system should be specified by developers. The top level Apache UIMA type is referred to as TOP, other primitive types include. int, String, Annotation and FSArray (an array of any annotations).

2.1.2 Component and Capability

UIMA components receive and update CAS one at a time. Each UIMA component has a *capability* property, which describes what *types* of annotations it takes as input and what *types* of annotations it may produce as output.

UIMA components can be deployed either locally, or remotely as SOAP web services. Remotely deployed web service components and locally deployed components can be freely combined in UIMA workflows.

2.1.3 Aggregate Component and Flow Controller

UIMA components can be either *primitive* or *aggregate*. Aggregate components include other components as subcomponents. Subcomponents may themselves be aggregate. In the case where an aggregate has multiple subcomponents these are by default processed in linear order. This ordering can be customized by implementing a custom *flow controller*.

2.2 U-Compare

U-Compare is a joint project of the University of Tokyo, the Center for Computational Pharmacology at the University of Colorado School of Medicine, and the UK National Centre for Text Mining.

U-Compare provides an integrated platform for users to construct, edit and compare workflows compatible with any UIMA component. It also provides a large, ready-to-use toolkit of interoperable NLP components for use with any UIMA based system. This toolkit is currently the world largest repository of type system compatible components. These all implement the U-Compare type system described in section 3.

[2] http://www.oasis-open.org/committees/uima/

[3] 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*.

2.2.1 Related Works

There also exist several other public UIMA component repositories: CMU UIMA component repository, BioNLP UIMA repository (Baumgartner et al., 2008), JCoRe (Hahn et al., 2008), Tsujii Lab Component Repository at the University of Tokyo (Kano et al., 2008a), etc. Each group uses their own type system, and so components provided by each group are incompatible. Unlike U-Compare these repositories are basically only collections of UIMA components, U-Compare goes further by providing a fully integrated set of UIMA tools and utilities.

2.2.2 Integrated Platform

U-Compare provides a variety of features as part of an integrated platform. The system can be launched with a single click in a web browser; all required libraries are downloaded and updated automatically in background.

The Workflow Manager GUI helps users to create workflows in an easy drag-and-drop fashion. Similarly, import/export of workflows, running of workflows and saving results can all be handled via a graphical interface.

U-Compare special parallel aggregate components allow combinations of specified components to be automatically combined and compared based on their I/O capabilities (Kano et al., 2008b). When workflows are run, U-Compare shows statistics and visualizations of results appropriate to the type of workflow. For example when workflows including parallel aggregate components are run comparison statistics between all possible parallel component combinations are given.

3 Integrated System for Pluggable Evaluation Metrics

While U-Compare already has a mechanism to automatically create possible combinations of components for comparison from a specified workflow, the comparison (evaluation) metric itself was hard coded into the system. Only comparison based on simple strict matching was possible.

However, many different evaluation metrics exist, even for the same type of annotations. For example, named entity recognition results are often evaluated based on several different annotation intersection criteria: exact match, left/right only match, overlap, etc. Evaluation metrics for nested components can be even more complex (e.g. biomedical relations, deep syntactic structures). Sometimes new metrics are also required for specific tasks. Thus, a mechanism for pluggable evaluation metrics in a standardized way is seen as desirable.

3.1 Pluggable Evaluation Component

Our design goal for the evaluation systems is to do as much of the required work as possible and to provide utilities to reduce developer's labor. We also want our design to be generic and fix within existing UIMA standards.

The essential process of evaluation can be generalized and decomposed as follows:

(a) prepare a pair of annotation sets which will be used for comparison,

(b) select annotations which should be included in the final evaluation step,

(c) compare selected annotations against each other and mark matched pairs.

For example, in the case of the Penn Treebank style syntactic bracket matching, these steps correspond to (a) prepare two sets of constituents and tokens, (b) select only the constituents (removing null elements if required), (c) compare constituents between the sets and return any matches.

In our new design, step (a) is performed by the system, (b) and (c) are performed by an evaluation component. The evaluation component is just a normal UIMA component, pluggable based on the UIMA standard. This component is run on a CAS which was constructed by the system during step (a). This CAS includes an instance of ComparisonSet type and its features GoldAnnotationGroup and TestAnnotationGroup. Corresponding to step (b), based on this input the comparison component should make a selection of annotations and store them as FSArray for both GoldAnnotations and TestAnnotations. Finally for step (c), the component should perform a matching and store the results as MatchedPair instances in the MatchedAnnotations feature of the ComparisonSet.

Precision, recall, and F1 scores are calculated by U-Compare based on the outputted ComparisonSet. These calculation can be overridden and customized if the developer so desires.

Implementation of the `compare()` method of the evaluation component is recommended. It is used by the system when showing instance based evaluations of what feature values are used in

matching, which features are matched, and which are not.

3.2 Combinatorial Evaluation and Error Analysis

By default, evaluation statistics are calculated by simply counting the numbers of gold, test, matched annotations in the returned ComparisonSet instance. Then precision, recall, and F1 scores for each CAS and for the complete set of CASes are calculated. Users can specify which evaluation metrics are used for each *type* of annotations based on the input specifications they set for supplied evaluation components.

Normally, precision, recall, and F1 scores are the only evaluation statistics used in the NLP community. It is often the case in many research reports that a new tool A performs better than another tool B, increasing the F1 score by 1%. In such cases it is important to analysis what proportion of annotations are shared between A, B, and the gold standard. Is A a strict 1% increase over B? Or does it cover 2% of instances B doesn't but miss a different 1%? Our system provides these statistics as well.

Further, our standardized evaluation system makes more advanced evaluation available. Since the evaluation metrics themselves are more or less arbitrary, we should carefully observe the results of evaluations. When two or more metrics are available for the same type of annotations, we can compare the results of each to analyze and validate the individual evaluations.

An immediate application of such comparison would be in a voting system, which takes the results of several tools as input and selects common overlapping annotations as output.

U-Compare also provides visualizations of evaluation results allowing instance-based error analysis.

4 U-Compare Type System

U-Compare currently provides the world largest set of type system compatible UIMA components. We will describe some of these in section 5. In creating compatible components in UIMA a key task is their *type system* definitions.

The U-Compare type system is designed in a hierarchical fashion with distinct types to achieve a high level of interoperability. It is intended to be a shared type system capable of mapping types originally defined as part of independent type systems (Kano et al., 2008c). In this section we describe the U-Compare type system in detail.

4.1 Basic Types

While most of the U-Compare types are inheriting a UIMA built-in type, Annotation (Figure 1), there are also types directly extending the TOP type; let us call these types as metadata types.

AnnotationMetadata holds a confidence value, which is common to all of the U-Compare annotation types as a feature of BaseAnnotation type. BaseAnnotation extends DiscontinuousAnnotation, in which fragmental annotations can be stored as a FSArray of Annotations, if any.

ExternalReference is another common metadata type where namespace and ID are stored, referring to an external ontology entity outside UIMA/U-Compare. Because it is not realistic to represent everything like such a detailed ontology hierarchy in a UIMA type system, this metadata is used to recover original information, which are not expressed as UIMA types. ReferenceAnnotation is another base annotation type, which holds an instance of this ExternalReference.

UniqueLabel is a special top level type for explicitly defined finite label sets, e.g. the Penn Treebank tagset. Each label in such a tagset is mapped to a single type where UniqueLabel as its

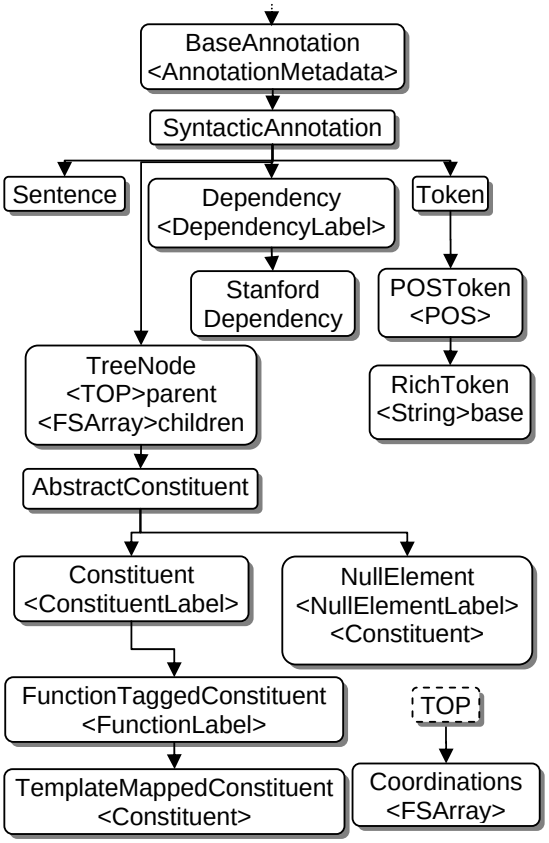

Figure 2. Syntactic Types in U-Compare.

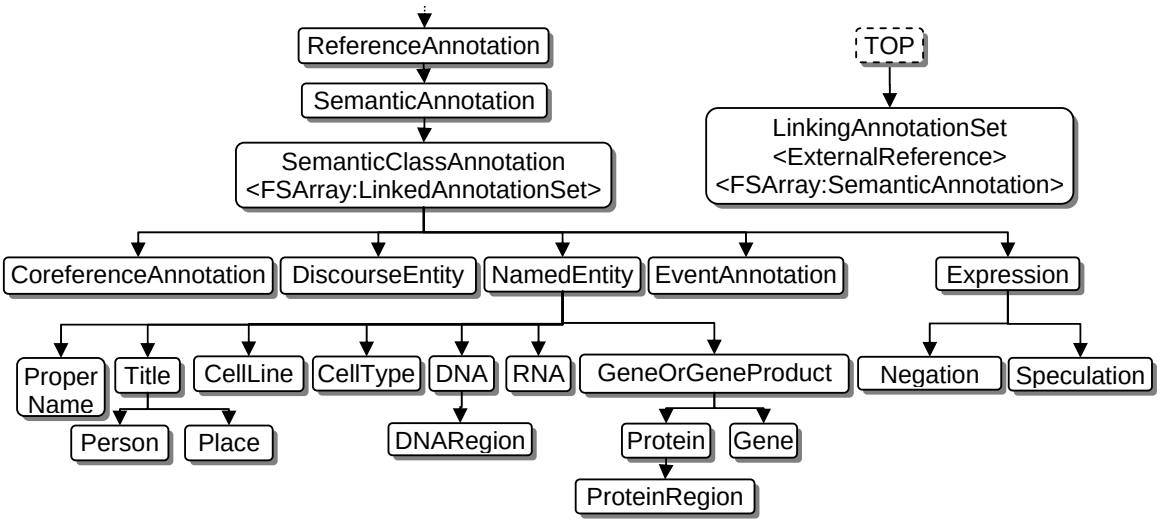

Figure 3. Semantic types in the U-Compare type system.

ancestor, putting middle level types if possible (e.g. Noun type for the Penn Treebank POS tagset). These types are omitted in the figure.

4.2 Syntactic Types

SyntacticAnnotation is the base type of all syntactic types (Figure 2). POSToken holds a POS label, RichToken additionally holds a base form. Dependency is used by dependency parsers, while TreeNode is for syntactic tree nodes. Constituent, NullElement, FunctionTaggedConstituent, TemplateMappedConstituent are designed to fully represent all of the Penn Treebank style annotations. Coordination is a set of references to coordinating nodes (currently used by the Genia Treebank). We are planning on extending the set of syntactic types to cover the outputs of several deep parsers.

4.3 Semantic Types

SemanticAnnotation is the base type for semantic annotations; it extends ReferenceAnnotation by holding the original reference.

SemanticClassAnnotation is a rather complex type designed to be somewhat general. In many cases, semantic annotations may reference other semantic annotations, e.g. references between biological events. Such references are often labeled with their roles which we express with the ExternalReference type. Such labeled references are expressed by LinkingAnnotationSet. As a role may refer to more than one annotation, LinkingAnnotationSet has an FSArray of SemanticAnnotation as a feature.

There are several biomedical types included in Figure 3, e.g. DNA, RNA, Protein, Gene, CellLine, CellType, etc. It is however difficult to decide which ontological entities should be included in such a type system. One reason for this is that such concepts are not always distinct; different ontologies may give overlapping definitions of these concepts. Further, the number of possible substance level entities is infinite; causing difficult in their expression as individual types. The current set of biomedical types in the U-Compare type system includes types which are frequently used for evaluation in the BioNLP research.

4.4 Document Types

DocumentAnnotation is the base type for document related annotations (Figure 4). It extends

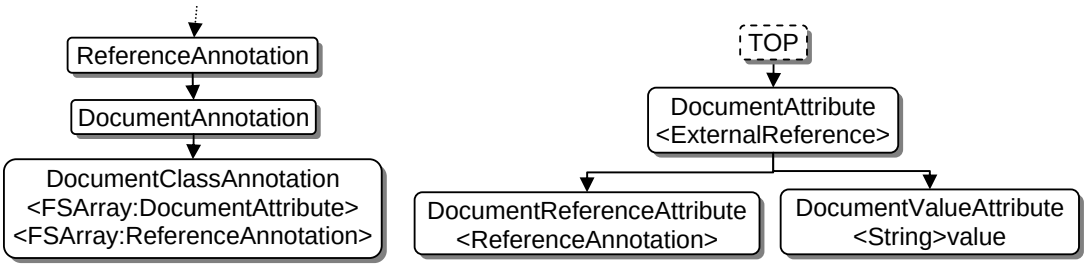

Figure 4. Document types in the U-Compare type system.

ReferenceAnnotation to reference the full external type in the same way as SemanticAnnotation.

DocumentClassAnnotation together with DocumentAttribute are intended to express XML style data. XML tags may have fields storing their values, and/or idref fields refering to other tags. DocumentValueAttiributerepresents simple value field, while DocumentReferenceAttribute represents idref type fields. A DocumentClassAnnotation corresponds to the tag itself.

Although these types can represent most document structures, we still plan to add several specific types such as Paragraph, Title, etc.

5 Interoperable Components and Utilities

In this section, we describe our extensive toolkit of interoperable components and the set of utilities integrated into the U-Compare system. All of the components in our toolkit are compatible with the U-Compare type system described in the previous section.

5.1 Corpus Reader Components

In the UIMA framework, a component which generates CASes is called a *Collection Reader*. We have developed several collection readers which read annotated corpora and generates annotations using the U-Compare type system.

Because our primary target domain was biomedical field, there are corpus readers for the biomedical corpora; Aimed corpus (Bunescu et al., 2006) reader and BioNLP '09 shared task format reader generate event annotations like protein-protein interaction annotations; Readers for BIO/IOB format, Bio1 corpus (Tateisi et al., 2000), BioCreative (Hirschman et al., 2004) task 1a format, BioIE corpus (Bies et al., 2005), NLPBA shared task dataset (Kim et al., 2004), Texas Corpus (Bunescu et al., 2005), Yapex Corpus (Kristofer Franzen et al., 2002), generate biomedical named entities, and Genia Treebank corpus (Tateisi et al., 2005) reader generates Penn Treebank (Marcus et al., 1993) style bracketing and part-of-speech annotations. Format readers require users to prepare annotated data, while others include corpora themselves, automatically downloaded as an archive on users' demand.

In addition, there is File System Collection Reader from Apache UIMA which reads files as plain text. We have developed an online interactive text reader, named Input Text Reader.

```
187
The document length in bytes is
output in the first line (end with
new line),
then the raw text follows as is
(attaching a new line in the end),
finally annotations follow line by
line.
0 187 Document id="u1"
0 3 POSToken id="u2" pos="DT"
....
```

Figure 5. An example of the U-Compare simple I/O format.

5.2 Analysis Engine Components

There are many tools covering from basic syntactic annotations to the biomedical annotations. Some of the tools are running as web services, but users can freely mix local services and web services.

For syntactic annotations: sentence detectors from GENIA, LingPipe, NaCTeM, OpenNLP and Apache UIMA; tokenizers from GENIA tagger (Tsuruoka et al., 2005), OpenNLP, Apache UIMA and Penn Bio Tokenizer; POS taggers from GENIA tagger, LingPipe, OpenNLP and Stepp Tagger; parsers from OpenNLP (CFG), Stanford Parser (dependency) (de Marneffe et al., 2006), Enju (HPSG) (Miyao et al., 2008).

For semantic annotations: ABNER (Settles, 2005) for NLPBA/BioCreative trained models, GENIA Tagger, NeMine, MedT-NER, LingPipe and OpenNLP NER, for named entity recognitions. Akane++ (Sætre et al., 2007) for protein-protein interaction detections.

5.3 Components for Developers

Although Apache UIMA provides APIs in both Java and C++ to help users develop UIMA components, a level of understanding of the UIMA framework is still required. Conversion of existing tools to the UIMA framework can also be difficult, particularly when they are written in other programming languages.

We have designed a simple I/O format to make it easy for developers who just want to provide a UIMA wrapper for existing tools.

Input of this format consists of two parts: raw text and annotations The first line of the raw text section is an integer of byte count of the length of the text. The raw text then follows with a newline character appended at the end. Annotations are then included; one annotation per line, sometimes referring another annotation by assigned ids (Figure 5). A line consists of begin position,

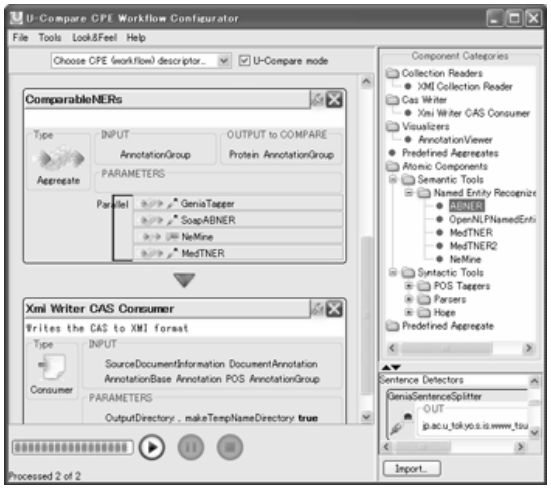

Figure 6. A screenshot of Workflow Manager GUI and Component Library.

end position, type name, unique id, and feature values if any. Double newlines indicates an end of a CAS.

Output of the component is lines of annotations if any created by the component.

U-Compare provides a wrapper component which uses this I/O format, communicating with wrapped tools via standard I/O streams.

5.4 Type System Converters

As U-Compare is a joint project, the U-Compare toolkit includes UIMA components originally developed using several different type systems. In order to integrate these components into the U-Compare type system, we have developed type system converter components for each external type system.

The CCP team at the University of Colorado made a converter between their CCP type system and our type system. We also developed converters for OpenNLP components and Apache UIMA components. These converters remove any original annotations not compatible with the U-Compare type system. This prevents duplicated converters from translating external annotation multiple times in the same workflow.

We are providing such non U-Compare components by aggregating with type system converters, so users do not need to aware of the type system conversions.

5.5 Utility Tools

We have developed and integrated several utility tools, especially GUI tools for usability and error analysis.

Figure 6 is showing our workflow manager GUI, which provides functions to create a user workflow by an easy drag-and-drop way. By clicking “Run Workflow” button in that manager window, statistics will be shown (Figure 8).

There are also a couple of annotation visualization tools. Figure 7 is showing a viewer for tree structures and HPSG feature structures. Figure 9 is showing a general annotation viewer, when annotations have complex inter-dependencies.

6 Summary and Future Directions

We have designed and developed a pluggable evaluation system based on the UIMA framework. This evaluation system is integrated with the U-Compare combinatorial comparison mechanism which makes evaluation of many factors available automatically.

Since the system behavior is dependent on the type system used, we have carefully designed the U-Compare type system to cover a broad range of concepts used in NLP applications. Based directly on this type system, or using type system converters, we have developed a large toolkit of type system compatible interoperable UIMA component. All of these features are integrated into U-Compare.

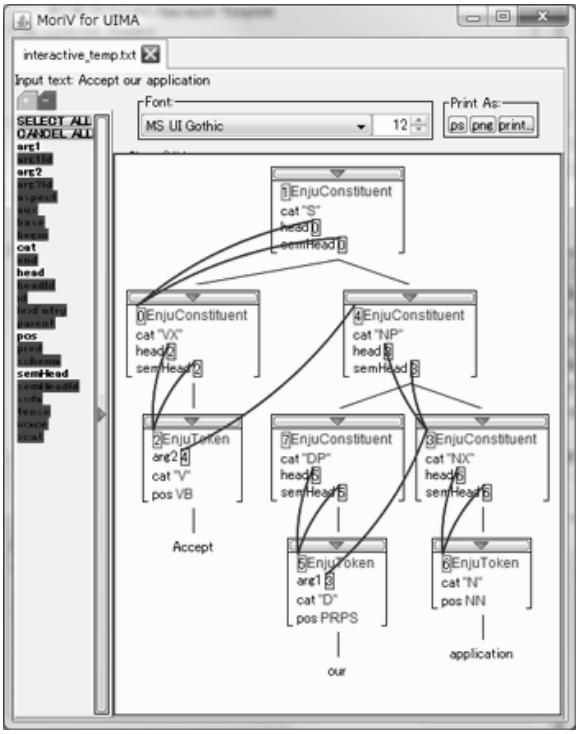

Figure 7. A screenshot of HPSG feature structure viewer, showing a skeleton CFG tree, feature values and head/semhead links.

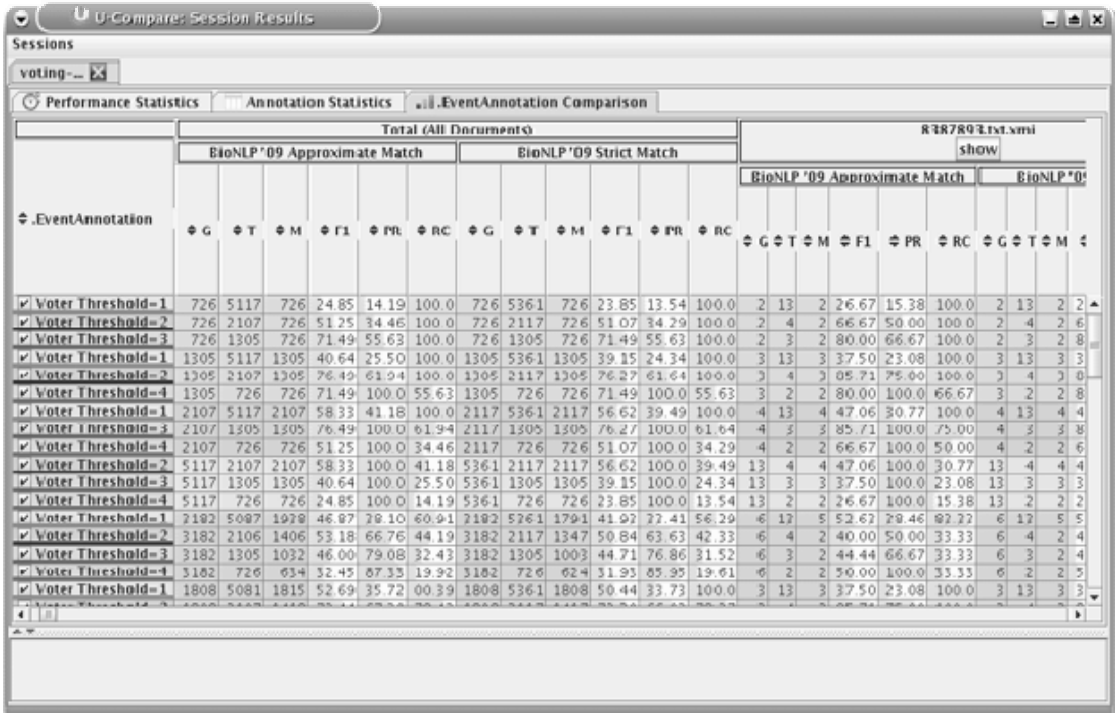

Figure 8. A screenshot of a comparison statistics showing number of instances (gold, test, and matched), F1, precision, and recall scores of two evaluation metrics on the same data.

In future we are planning to increase the number of components available, e.g. more syntactic parsers, corpus readers, and resources for languages other than English. This will also required enhancements to the existing type system to support additional components. Finally we also hope to add integration with machine learning tools in the near future.

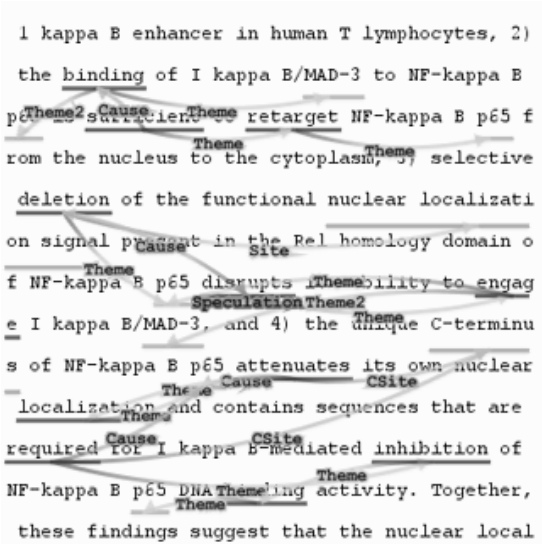

Figure 9. A screenshot of a visualization of complex annotations.

Acknowledgments

We wish to thank Dr. Lawrence Hunter's text mining group at Center for Computational Pharmacology, University of Colorado School of Medicine, for helping build the type system and for making their tools available for this research. This work was partially supported by Grant-in-Aid for Specially Promoted Research (MEXT, Japan). The National Centre for Text Mining is funded by JISC.

References

W. A. Baumgartner, Jr., K. B. Cohen, and L. Hunter. 2008. *An open-source framework for large-scale, flexible evaluation of biomedical text mining systems*. J Biomed Discov Collab, 3(1), 1.

Ann Bies, Seth Kulick, and Mark Mandel. 2005. *Parallel entity and treebank annotation.* In Proceedings of the the Workshop on Frontiers in Corpus Annotations II: Pie in the Sky, ACL, Ann Arbor, Michigan, USA.

Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun Kumar Ramani, et al. 2005. *Comparative experiments on learning information extractors for proteins and their interactions*. Artificial Intelligence in Medicine, 33(2), 139-155.

Razvan Bunescu, and Raymond Mooney. 2006. *Subsequence Kernels for Relation Extraction*. In Y. Weiss, B. Scholkopf and J. Platt (Eds.), *Advances in Neural Information Processing Systems 18* (171--178). Cambridge, MA: MIT Press.

Marie-Catherine de Marneffe, Bill MacCartney, and Christopher D. Manning. 2006. *Generating typed dependency parses from phrase structure parses.* In Proceedings of the the 5th International Conference on Language Resources and Evaluation (LREC 2006).

David Ferrucci, and Adam Lally. 2004. *Building an example application with the Unstructured Information Management Architecture*. Ibm Systems Journal, 43(3), 455-475.

David Ferrucci, Adam Lally, Daniel Gruhl, and Edward Epstein. 2006. *Towards an Interoperability Standard for Text and Multi-Modal Analytics.*

U. Hahn, E. Buyko, R. Landefeld, M. Mühlhausen, M. Poprat, K. Tomanek, et al. 2008, May. *An Overview of JCoRe, the JULIE Lab UIMA Component Repository.* In Proceedings of the LREC'08 Workshop, Towards Enhanced Interoperability for Large HLT Systems: UIMA for NLP, Marrakech, Morocco.

Lynette Hirschman, Alexander Yeh, Christian Blaschke, and Antonio Valencia. 2004. *Overview of BioCreAtIvE: critical assessment of information extraction for biology*. BMC Bionformatics, 6(Suppl 1:S1).

Yoshinobu Kano, William A Baumgartner, Luke McCrohon, Sophia Ananiadou, Kevin B Cohen, Lawrence Hunter, et al. 2009. *U-Compare: share and compare text mining tools with UIMA.* Bioinformatics, accepted.

Yoshinobu Kano, Ngan Nguyen, Rune Sætre, Keiichiro Fukamachi, Kazuhiro Yoshida, Yusuke Miyao, et al. 2008c, January. *Sharable type system design for tool inter-operability and combinatorial comparison.* In Proceedings of the the First International Conference on Global Interoperability for Language Resources (ICGL), Hong Kong.

Yoshinobu Kano, Ngan Nguyen, Rune Sætre, Kazuhiro Yoshida, Keiichiro Fukamachi, Yusuke Miyao, et al. 2008b, January. *Towards Data And Goal Oriented Analysis: Tool Inter-Operability And Combinatorial Comparison.* In Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP), Hyderabad, India.

Yoshinobu Kano, Ngan Nguyen, Rune Sætre, Kazuhiro Yoshida, Yusuke Miyao, Yoshimasa Tsuruoka, et al. 2008a, January. *Filling the gaps between tools and users: a tool comparator, using protein-protein interaction as an example.* In Proceedings of the Pacific Symposium on Biocomputing (PSB), Hawaii, USA.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. *Introduction to the Bio-Entity Recognition Task at JNLPBA.* In Proceedings of the International Workshop on Natural Language Processing in Biomedicine and its Applications (JNLPBA-04), Geneva, Switzerland.

Kristofer Franzen, Gunnar Eriksson, Fredrik Olsson, Lars Asker, Per Liden, and Joakim Coster. 2002. *Protein names and how to find them*. International Journal of Medical Informatics, 67(1-3), 49-61.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. *Building a large annotated corpus of English: the penn treebank*. Computational Linguistics, 19(2), 313-330.

Yusuke Miyao, and Jun'ichi Tsujii. 2008. *Feature Forest Models for Probabilistic HPSG Parsing*. Computational Linguistics, 34(1), 35-80.

Rune Sætre, Kazuhiro Yoshida, Akane Yakushiji, Yusuke Miyao, Yuichiro Matsubayashi, and Tomoko Ohta. 2007, April. *AKANE System: Protein-Protein Interaction Pairs in BioCreAtIvE2 Challenge, PPI-IPS subtask.* In Proceedings of the Second BioCreative Challenge Evaluation Workshop.

Burr Settles. 2005. *ABNER: an open source tool for automatically tagging genes, proteins and other entity names in text*. Bioinformatics, 21(14), 3191-3192.

Yuka Tateisi, Tomoko Ohta, Nigel Collier, Chikashi Nobata, and Jun'ichi Tsujii. 2000, August. *Building an Annotated Corpus from Biology Research Papers.* In Proceedings of the COLING 2000 Workshop on Semantic Annotation and Intelligent Content, Luxembourg.

Yuka Tateisi, Akane Yakushiji, Tomoko Ohta, and Jun'ichi Tsujii. 2005, October. *Syntax Annotation for the GENIA Corpus.* In Proceedings of the the Second International Joint Conference on Natural Language Processing (IJCNLP '05), Companion volume, Jeju Island, Korea.

Yoshimasa Tsuruoka, Yuka Tateishi, Jin Dong Kim, Tomoko Ohta, J. McNaught, Sophia Ananiadou, et al. 2005. *Developing a robust part-of-speech tagger for biomedical text.* In *Advances in Informatics, Proceedings* (Vol. 3746, 382-392). Berlin: Springer-Verlag Berlin.