Integrating Annotation Tools into UIMA for Interoperability

Scott Piao, Sophia Ananiadou and John McNaught

School of Computer Science & National Centre for Text Mining
The University of Manchester
UK

{scott.piao;sophia.ananiadou;john.mcnaught}@manchester.ac.uk

Abstract

In this paper, we discuss the issue of implementing the interoperability of natural language annotation tools for text mining with the Unstructured Information Management Architecture (UIMA) (Ferrucci and Lally, 2004; http://incubator.apache.org/uima). In particular, we discuss the practical issue of designing UIMA annotation schemes for text mining applications based on our experience in the EC BOOTStrep Project. Currently, a major obstacle to the efficient integration of existing annotation tools for text mining is their lack of interoperability. Many such tools have been developed in different projects following different theoretical approaches and annotation schemes, hence they are highly specialized. Consequently, much effort and time are being wasted in adjusting and modifying existing tools to make them interoperable, or even worse, in "re-inventing the wheel". Our experience shows that UIMA provides a practical way of implementing interoperability for tools, but there are some practical issues to be tackled. Particularly, we found it a challenging task to design a common/shared UIMA annotation scheme for different sets of tools. A practical and flexible approach to solve such problems is required.

1. Introduction

Over the past few decades, we have seen an ever increasing number of natural language annotation tools designed for carrying out a variety of tasks such as tokenization, parsing, named entity recognition, semantic annotation, etc. Such a rich pool of annotation tools should allow for rapid development of information management systems via composing the existing tools. In reality, however, it can be a complicated task to integrate tools of different origins. Because such tools are often developed in different projects following different theoretical guidelines and implementing specific annotation schemes, these tools tend to be highly specialized and are not compatible with each other thus lack interoperability.

This lack of interoperability of annotation tools becomes a stumbling block when we need to integrate them into larger systems. A typical scenario is to develop a semantic analysis system based on existing annotation tools. In order to carry out semantic analysis of the unstructured information contained in natural language text, we need to integrate a set of annotation tools to form a workflow, which can extract information into certain structured forms. For instance, in order to extract relations between named entities (NEs) from texts, we need to combine tools including sentence delimiter, tokeniser, part of speech (POS) tagger, chunker, syntactic parser, NE identifier, co-reference detector etc. If the tools and the annotation schemes available for such tasks are not interoperable with each other, much time and effort can be wasted in "cutting and fitting" them, or even worse, in "re-inventing the wheel". Note that, in some cases, integrating incompatible tools may involve a prohibitive amount of technical work.

In recent years, the interoperability issue of annotation tools has received increasing attention, and some systems have been designed and developed to address this problem. For example, the General Architecture for Text Engineering (GATE) (Cunningham *et al*. 2002) provides a set of compatible annotation tools and resources in the form of a class library (SDK) with a graphical development environment. Another major work in this regard is the SciBorg Project (Copestake et al., 2006)[1],

[1] Also see website: http://www.cl.cam.ac.uk/~aac10/escience/sciborg.html

in which a new mark-up language and a system are being developed to combine and integrate the information produced by various annotation tools for extracting knowledge from Chemistry papers. The ATLAS (Laprun *et al*., 2002; http://www.nist.gov/speech/atlas/) project had a similar goal.

However, compared to other existing systems, the Unstructured Information Management Architecture (UIMA) (Ferrucci and Lally, 2004) provides a more flexible and extensible architecture for implementing interoperability. It is a prominent recent development in the area of information management, aiming to provide "a software architecture for defining and composing interoperable text and multimodal analytics" (http://incubator.apache.org/uima). (Here analytic refers to an analysis component, analysis service or their combination.) This architecture is adopted in the EC BOOTStrep Project[2] to support the interoperability of various annotation tools involved in the project. In this paper, we focus on some practical issues concerning the application of UIMA to text mining tasks based on our experiences in this project.

The remainder of this paper is organized as follows. Section 2 briefly describes the UIMA architecture, section 3 discusses the issue of UIMA annotation schemes, section 4 reports our work on a UIMA annotation scheme as a case study, section 5 provides a brief survey of related work, and section 6 concludes the paper.

2. UIMA, an architecture for interoperability

As mentioned already, UIMA provides an architecture supporting interoperability of analysis components, or analytics (we are only concerned with text analytics in this paper). In this architecture, UIMA analytics share data and analysis results. A common data structure named the Common Analysis Structure (CAS) is used to represent the analysis data, including the artefact (data to be analyzed) and its metadata (data about the artefact). A CAS instance contains objects linked by an object graph. Each object, which represents a data structure, is defined by a set of properties implemented as slots. For example, an object of class Token can be defined by the properties of lemma, POS tag(s) and a pair of regional references pointing to the beginning and ending positions of the token. In addition, UIMA facilitates representation and separate processing of different views of the same artefact, such as the plain text and HTML version of a web page, translations of a document, etc.

In UIMA, the annotation information is represented with classes, or types (type is used henceforth). A type is defined by a set of features, which can be either primitive types such as String, Integer or references to other types. For example, a *Term* type may contain attributes such as *baseForm*, *abbreviation*, *confidenceScore* (if extracted with a statistical tool) etc. UIMA defines a set of primitive types (e.g. String, Double, etc.) and base types (e.g. FSArray, Annotation, etc.), but it crucially does not provide any standard type system. It is the users' responsibility to define the type system(s) suitable for their own needs.

A text analytic typically describes or classifies certain regions of a text according to pre-defined categories, such as types of syntactic constituents, types of NEs, relations between NEs, etc. UIMA adopts a stand-off annotation model, in which an annotation type object typically points to a text region, such as a word or person name. This supports a flexible multilayer annotation model which allows overlapping or even conflicting annotations of the same text region. UIMA provides a built-in type, named *Annotation*, which provides the parent type for the user-defined types.

In order to support interoperability, all of the objects contained in a CAS must be instances of either the built-in UIMA types or those pre-defined in a type system(s). Each UIMA analytic can access the CAS, process objects in it, update existing metadata, create new objects in the CAS, or create new CAS(es). In this process, the type system functions as a common language among the analytics.

There are two important aspects of interoperability to be considered. The first is interoperability between the analytics in a work flow. The input and output types of the analytics must be within the range of the given type system. This facilitates the exchange of data and metadata between the analytics..For instance, for a POS tagger and a syntactic parser to interoperate, the tagger must output token types with POS properties that are compatible with those expected by the parser, i.e. both must use an agreed tagset. The other important aspect is interoperability between similar analytics. In this regard, those analytics performing the same functions accept as inputs and produce as outputs the same types that are pre-defined by a type system. For example, if a UIMA package

[2]The BOOTStrep Project is funded by the EC´s 6th Framework Programme, aiming to pull together existing biological databases and various terminological repositories and implement a text analysis system to populate a Bio-Lexicon and a Bio-Ontology to support text mining. For further details, see: www.bootstrep.org

contains two POS taggers, they work on the same input types and produce the same types. This makes them inter-substitutable and their performances can be compared with convenience. This is important for rapidly identifying the optimal tools for a given task.

Externally developed annotation tools can be wrapped to integrate and interoperate within the UIMA architecture, as illustrated by Fig. 1. UIMA provides interfaces with which the annotation information produced by these tools can be converted into UIMA types and features. It also supports interoperability at various levels, including the data level, programming model level, services level, etc.[3]

As shown above, UIMA provides a sophisticated software architecture for achieving interoperability of annotation tools.

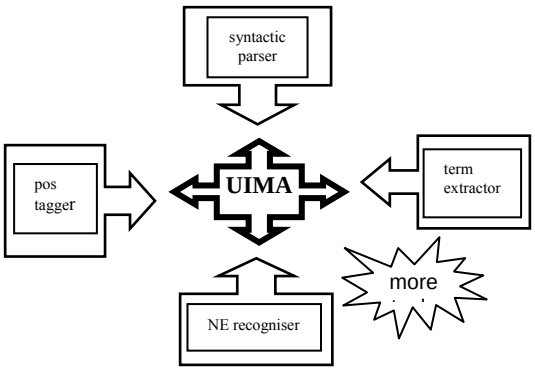

Fig. 1: Integration of annotation tools with UIMA

3. Some practical issues in applying UIMA in text mining

Text mining technology enables us to "collect, maintain, interpret, curate and discover knowledge" (Ananiadou and McNaught, 2006). To achieve this goal, we need to integrate a set of NLP tools to form a text mining work flow. As we discussed previously, UIMA provides asoftware architecture for building such applications based on existing NLP tools. However, before we can efficiently apply the UIMA architecture for text mining purposes, there are some practical issues to be resolved. In the following sections, we discuss some of these issues based on our practical experience.

3.1 UIMA annotation scheme

A major issue regarding the application of UIMA in text mining is the adoption or design of a common or shared annotation scheme for a project or research community. As UIMA does not provide its own standard annotation scheme, or type system, we need to adopt or design an annotation scheme that caters for our specific requirements.

In our case, where text mining application is concerned, we need to design an annotation scheme that generally meets the needs of the TM community, at least for certain groups sharing specific research interests and software requirements. In particular, if several tool sets of different origins are involved, a common UIMA annotation scheme becomes indispensable.

However, it should be stressed that we are not aiming at a universal standard annotation scheme for NLP and text mining. It can be tempting to develop such a standard, whereas it is not practical in reality[4]. This is due to the fact that many existing annotation tools are based on different theories and specific annotation schemes. It is a well known fact that any annotation scheme has its limitation in terms of standardness. In his proposal of maxims regarding corpus annotation, Leech (1997: 6-7) points out: "*… the annotation scheme is made available to a research community on a caveat emptor principle. It does not come with any 'gold standard' guarantee, but is offered as a matter of practical usefulness only … their goal should be to adopt annotations which are as widely accepted and understood as can be managed … No one annotation scheme should claim authority as an absolute standard. Annotation schemes tend to vary for good practical reasons*". This would be even more true if multiple languages are involved.

For instance, the Lancaster CLAWS POS tagger (Garside, 1987) and the TreeTagger for English[5] employ different POS categories: *Cx* and *PTB* tagsets respectively (Manning and Schütze, 1999: 139-143). Similarly, MST Parser (McDonald et al. 2005) is based on dependency grammar (Hudson, 1984) whereas the Enju parser (Miyao and Tsujii, 2005) is based on Head-driven Phrase Structure Grammar (HPSG) (Pollard and Sag, 1994). None of them can claim to be a gold standard, and it is also difficult to merge all of them into a single standard.

Meanwhile, for the practical purposes of data exchange, tool compatibility and reusability, etc. within certain projects or a research community, we need a common

[3] For further details, see http://incubator.apache.org/uima/documentation.html

[4] Although there are major related consensus-building initiatives such as those of the EC LIRICS project and ISO TC37/SC4.

[5] For detailed information about TreeTagger, see http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger.

annotation scheme, with the pre-condition that it is flexible enough to accommodate the different features of the individual annotation schemes associated with the tools being used. Quoting Leech (1997) again: "*If different researchers need to interchange data and resources (such as annotated corpora), this is more easily achieved if the same standards or guidelines have been applied in different centres. The need for some kind of standardization of annotation practices is particularly evident when we come to the mutual exchange of corpus software utilities … But the need is to encourage convergent practice without imposing a straitjacket of uniformity which could inhibit flexibility and productive innovation*". Therefore, an important issue in designing such a common annotation scheme is to keep a balance between standardization and flexibility.

It is a relatively trivial task to achieve interoperability between tools producing lower–level annotations, such as text structural mark-up, tokenization, POS tagging, etc. For instance, tokenizing tools and sentence breakers generally produce the same or similar outputs which can be easily mapped to each other. Very often there are only superficial differences, such as names of types or features. Although POS taggers present more challenges, in many cases they can be mapped quite well, although the mapping is not always bi-directional. For example, the Lancaster C5/7 tagset can be neatly mapped to the Penn Treebank (Marcus et al., 1993) tagset, although the reverse is difficult. At least generally approximate mappings are possible for such annotations.

However, the higher level annotations, including syntactic parsing and semantic annotation, can present tough challenges. For example, it is difficult to map between the outputs of syntactic parsers based on dependency grammar (DG) and HPSG. Similar difficulties can be expected between semantic annotation schemes which are based on different semantic categories, e.g. between the Lancaster UCREL semantic lexicon (Piao *et al*., 2006) and WordNet (Fellbaum, 1998).

3.2 Solutions to integration of different annotation schemes in UIMA

There are three common ways of dealing with the discrepancies between annotation schemes under the UIMA framework.

The first approach is a minimalist approach, in which the UIMA scheme includes only the elements shared between the different schemes under consideration, or the intersection of the different schemes.

$$uima\_scheme = scheme_1 \cap scheme_2 \cap \ldots \cap scheme_n \quad (n >= 2).$$

For example, let the three circles in Fig. 2 denote three different annotation schemes. According to the minimalist approach, the UIMA scheme is denoted by the overlapping area of the three circles. With such an approach, it is guaranteed that the information needed for populating the types and features of the shared UIMA annotation scheme can always be obtained from the annotation tools involved. Therefore, the developers of UIMA analytics can make full use of the types and features of the shared scheme, and can rest assured that they are always available. The downside aspect of this approach is the limitation of the information that can be represented by the UIMA scheme, for we would lose the information of the unshared parts of the schemes.

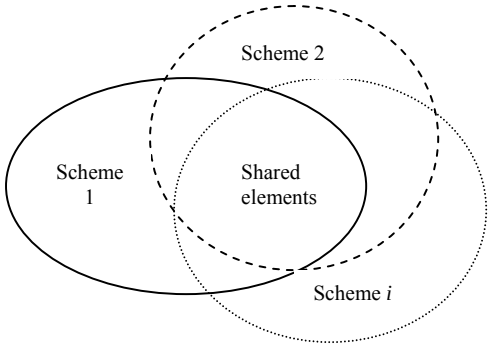

Fig. 2: Multiple annotation schemes in UIMA.

The opposite approach can be a maximalist approach, in which the shared UIMA annotation scheme is the union of the elements available from all of the different schemes. In Fig. 2, this would be the entire area comprising the three circles.

$$uima\_scheme = scheme_1 \cup scheme_2 \cup \ldots \cup scheme_n$$

In this approach, the unshared elements have to be set as optional types and features, and the availability of their values would depend on which tool(s) are included in the workflow. An obvious benefit of such an approach is the maximized availability of information, i.e. it does not lose any information that can potentially be produced by the annotation tools involved. With UIMA's stand-off annotation model, overlapping or conflicting annotations can be stored as multi-layer annotations. The developers of UIMA analytics can potentially make use of all of the information that can be produced/encoded by various annotation tools/schemes. However, the potential problem

is that the values of some types and features may not be available unless all of the tools involved are run in parallel, which may not be desirable, or may even be infeasible in certain circumstances. Consequently, the developer would have to check the availability of the values of types and features when they implement analytics based on the UIMA scheme being used. Moreover, it would be difficult to carry out comparative evaluations of different tools designed for the same functionality, e.g. different POS taggers, as the annotation results can be encoded with incompatible annotation schemes and thus lack comparability.

The third approach is to design a neutral common UIMA scheme, or a standard scheme for a given project or community, then map all other schemes into this standard. In theory, this would be an ideal solution, in which all existing different annotation schemes can be mapped into a standard single scheme and the interoperability of the schemes and tools can be guaranteed. Wherever possible, this goal should be pursued. Nevertheless, in many practical situations it is difficult, if not impossible, to achieve. As we pointed out, no annotation scheme can claim to be the sole standard, and it is very difficult to reach a unanimous agreement on any such standard, even among a small number of project partners.

In practice, we most likely need to adopt a hybrid approach. Very often, a UIMA application can contain a mixture of annotation schemes and tools of different origins, some of which can be identical or similar whereas the others can be incompatible and difficult to merge into a single standard. In such situations, the only practical approach can be to merge the schemes as much as possible while leaving the difficult parts as parallel paths of the UIMA scheme using the stand-off annotation mechanism.

4. A case study

The BOOTStrep project, which involves integrating existing NLP tools for knowledge harvesting and text mining in the biology domain, is a typical case concerning a shared UIMA annotation scheme.

One of the tasks ofthis project is to design a shared annotation scheme for the interoperability of a collection of annotation tools under the UIMA framework. The tools involved in the project have been developed in different projects at different research institutes. Among these are the Genia tagger (Tsuruoka et al., 2005), the Enju parser (Miyao and Tsujii, 2005) , a co-reference tool (Yang et al., 2004a, 2004b), and the OpenNLP tools[6].

While those tools for low-level annotations are largely compatible, including sentence breakers, morphological analyzers and POS taggers, the full syntactic parsers present a tough challenge to the integration work due to their different grammatical models. For example, the OpenNLP parser trained on the Penn Treebank corpus produces phrase structure parse trees while the Enju parser modelled on HPSG produces predicate argument structures as the primary output. Even in its secondary phrase structure output, each phrase has distinct features as shown in Fig. 3 below.

$$phrase\_feature = \begin{cases} offset(begin) \\ offset(end) \\ category \\ syntactic\_head \\ semantic\_head \\ parent \end{cases}$$

Fig. 3: Main features of phrase type of the Enju parser.

Such a discrepancy between the outputs of different syntactic parsers causes difficulty for merging/mapping them into a single common annotation scheme. As a result, in the initial stages of our project, a pair of parallel sub-schemes was designed in order to accommodate distinct features of the different tool sets.

Subsequent experimentation has revealed that it is feasible to merge these two sub-schemes in many parts, including text structural marking-up, tokenizing, morpho-syntactic analysis, etc. The most difficult part found so far concerns full syntactic parsing. We envisage that this would be the case for many similar projects involving the interoperability issue of annotation tools. Therefore, we conclude that the practical way of integrating the different sets of schemes and tools is to design a partially divergent UIMA scheme, as illustrated in Fig. 4.

As one can expect, the divergent part of such a shared annotation scheme may cause problems and confusion to UIMA analytics developers and therefore it needs special attention. One of the possible technical solutions to this problem can be the UIMA View mechanism which can be used to represent subsets of objects in a UIMA CAS. For example, the UIMA types pertaining to the divergent parts of the shared scheme (refer to Fig. 4) can be grouped into separate views, linking them with a specific

[6] For details of OpenNLP tools, see http://opennlp.sourceforge.net/projects.html

parser. This would allow the UIMA analytics developers to easily identify and access those types related to the specific parser he activates.

As our experience shows, it is a challenging task to design/develop a common UIMA annotation scheme for UIMA applications when various annotation tools are involved. Although a single standard UIMA annotation scheme is desirable, in practical circumstances it is often necessary to take flexible approaches to accommodate the distinct features of individual schemes which are difficult to map or merge.

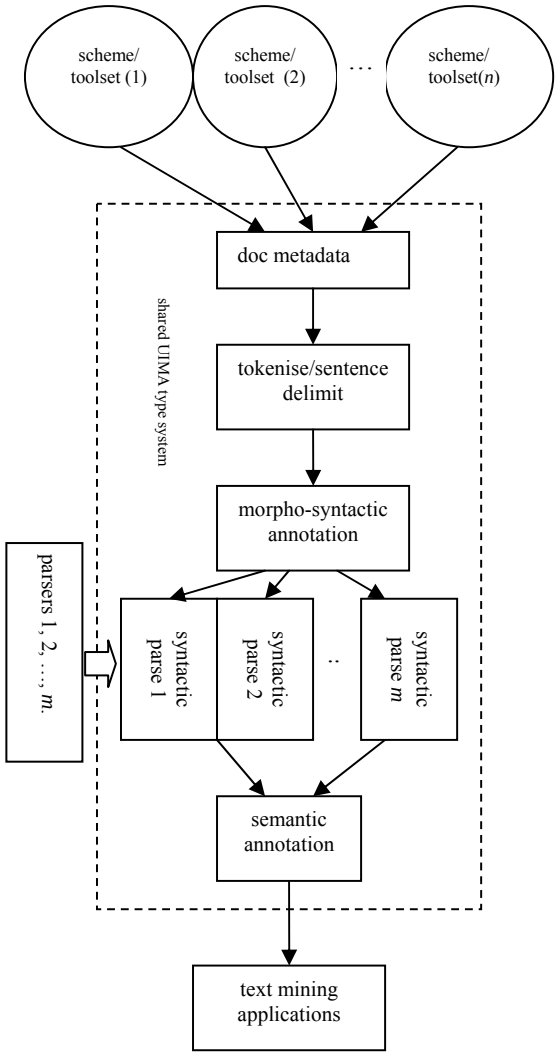

Fig. 4: A partially divergent UIMA annotation scheme.

Another issue we face concerns the level at which interoperability takes place. In the UIMA architecture, we can implement interoperability at various levels, including the data level, programming model level, services level, etc. If the tools are implemented with the same programming language as that of the UIMA SDK (currently Java), or the tools are locally available, we can implement interoperability at the programming model level, i.e. directly integrate the tools via wrappers into the UIMA system and put them into the workflow. However, for various reasons, such as copyright problems, lack of portability etc, sometimes it is difficult to collect all the tools together at a single site. In such cases, the solution is to build a Web Services based distributed UIMA system. UIMA provides a means of implementing analysis services, such as SOAP (Simple Object Access Protocol/Service Oriented Architecture Protocol) services, which remotely interoperate over the network. We adopted this approach to achieve interoperability between NLP tools which are located in different institutes, as illustrated by Fig. 5. Although the stability, scalability and efficiency of this approach remain to be fully examined, our initial tests show that it can potentially provide a good solution for integrating tools distributed across different locations.

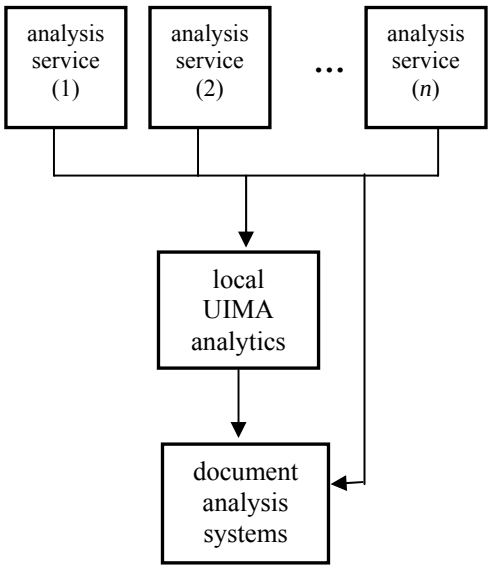

Fig. 5: Distributed UIMA system

5. Related work

Besides the GATE and SciBorg projects already mentioned, numerous projects have been carried out aiming at the interoperability of annotation tools. Much of the past work focuses on the development and designing of standard and shared annotation schemes. Some of these have been widely accepted as quasi-standards in certain communities.

Major annotation guidelines and schemes include EAGLES/ISLE[7] TEI (The Text Encoding Initiative)[8] , XCES (Corpus Encoding Standard for XML)[9], Penn TreeBank[10], Dublin Core Metadata Initiative[11], LAF (Linguistic Annotation Framework) (Ide and Romary, 2004), etc. They lay guidelines for encoding information stored in natural language text, which is to be followed by annotator developers.

Today, there are ongoing efforts to set out such annotation standards. For example, ISO/TC 37/SC4 is working towards setting international standards for annotation and processing of language resources[12]. Similar efforts are being made by the EC LIRICS project. With respect to the biomedical text mining area, Genia (Kim et al., 2003) and PennBioIE annotation schemes[13] are among the most influential annotation schemes.

While these annotation schemes provide guidelines to support data sharing and compatibility of annotation tools to a certain extent, a unified practical software architecture is required for implementing interoperability among annotation tools. UIMA fills this gap by providing a flexible and practical means of implementing interoperability based on existing annotation tools and annotation schemes.

6. Conclusion

In this paper, we discussed some practical issues pertaining to the implementation of interoperability of annotation tools under the UIMA architecture. In our experience, UIMA can potentially provide a flexible and efficient platform to support and facilitate interoperability. Nevertheless, there are some practical issues to be resolved before we can efficiently apply UIMA to practical TM applications. A particularly challenging issue in this regard is the designing of a common UIMA annotation scheme, or a shared UIMA type system, for tools of different origins. As UIMA continually evolves and becomes more sophisticated, we will further explore the issue of its application to text mining.

[7] See http://www.ilc.cnr.it/EAGLES96/isle/ISLE_Home_Page.htm

[8] See http://www.tei-c.org/

[9] See http://www.cs.vassar.edu/XCES/

[10] See http://www.cis.upenn.edu/~treebank/

[11] See http://dublincore.org/

[12] See *http://www.tc37sc4.org/new_doc/ISO_TC_37_SC4_N311_Linguistic%20Annotation%20Framework.pdf*

[13] See http://bioie.ldc.upenn.edu/

Acknowledgement

This work is supported by the EC BOOTStrep Project (Ref. FP6 - 028099) and the JISC/BBSRC/EPSRC funded UK National Centre for Text Mining (www.nactem.ac.uk). We thank BOOTStrep project partners for their co-operation in the project work related to this paper.

References:

Ananiadou, S., and J. McNaught (eds). 2006. *Text Mining for Biology and Biomedicine*. Boston, MA: Artech House.

Copestake, A., P. Corbett, P. Murray-Rust, CJ Rupp, A. Siddharthan, S. Teufel and B. Waldron. 2006. An architecture for language processing for scientific texts. *Proceedings of the UK e-Science All Hands Meeting 2006*. Nottingham, UK.

Cunningham, H., D. Maynard, K. Bontcheva and V. Tablan. 2002. 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 (ACL'02)*, Philadelphia, US.

Fellbaum, C. (ed.). 1998. *WordNet: An electronic lexical database*. Cambridge, MA: MIT Press.

Ferrucci, D. and A. Lally. 2004. Building an example application with the Unstructured Information Management Architecture, *IBM Systems Journal* 43(3): 455-475.

Garside, R. 1987. The CLAWS Word-tagging System. In: R. Garside, G. Leech and G. Sampson (eds), *The Computational Analysis of English: A Corpus-based Approach*. London: Longman.

Hudson, R. 1984. *Word Grammar*. Blackwell.

Ide, N. and L. Romary. 2004. International standard for a linguistic annotation framework. *Journal of Naturaql Language Engineering*, 10:3-4. pp. 211-225.

Kim, J.D., T. Ohta, Y. Teteisi and J. Tsujii. 2003. GENIA corpus - a semantically annotated corpus for bio-textmining. *Bioinformatics* 19(suppl. 1). pp. i180-i182.

Leech, G. 1997. Introducing Corpus Annotation. In Garside, Leech and McEnery (eds.) *Corpus Annotation – Linguistics Information from Computer Text Corpora*, pp. 1-18.

Marcus, M.P., B. Santorini and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The PENN TREEBANK. *Computational Linguistics*, 19(2):313–330.

McDonald, R., F. Pereira, K. Ribarov and J. Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, Vancouver, British Columbia, Canada. pp. 523 – 530.

Miyao, Y. and J. Tsujii. 2005. Probabilistic Disambiguation Models for Wide-Coverage HPSG Parsing. *In Proceedings of ACL-2005*, Ann Arbor, Michigan. pp. 83-90.

Piao, S., D. Archer, O. Mudraya, P. Rayson, R. Garside, A. M. McEnery and A. Wilson. 2006. A large semantic lexicon for corpus annotation. *In Proceedings from the Corpus Linguistics Conference Series on-line e-journal* 1(1), ISSN 1747-9398.

Pollard, C. and I. Sag. 1994. *Head-Driven Phrase Structure Grammar*. Chicago: University of Chicago Press.

Tsuruoka, Y., Y. Tateishi, J.D. Kim, T. Ohta, J. McNaught, S. Ananiadou and J. Tsujii. 2005. Developing a Robust Part-of-Speech Tagger for Biomedical Text, *Advances in Informatics - 10th Panhellenic Conference on Informatics*, LNCS 3746, pp. 382-392.

Yang, X., J. Su, G. Zhou and C. L. Tan. 2004a. An NP-Cluster approach to coreference resolution, In *Proceedings of the 20th International Conference on Computational Linguistics (COLING04)*, Geneva, Switzerland.

Yang, X., J. Su, G. Zhou and C. L. Tan. 2004b. Improving pronoun resolution by incorporating coreferential information of candidates. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL04)*, Barcelona, Spain. pp. 127-134.