A framework for representing and managing linguistic annotations based on typed feature structures

X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K. Gojenola* G. Labaka A. Sologaistoa A. Soroa
Faculty of Computer Science, Donostia / *School of Engineering, Bilbo
University of the Basque Country (UPV/EHU)
The Basque Country
jipdisaa@si.ehu.es

Abstract

In this paper we present a framework for dealing with linguistic annotations. Our aim is to establish a flexible and extensible infrastructure which follows a coherent and general representation scheme. This proposal provides us with a well-formalized basis for the exchange of linguistic information. We use TEI-P4 conformant feature structures as a representation schema for linguistic analyses. We have identified the consistent underlying data model which captures the structure and relations contained in the information to be manipulated. This data model has been represented by classes following the object-oriented paradigm. The huge amount of information generated is stored in an XML database that provides fast answers to common queries. With the aim of helping users to manipulate linguistic annotations generated by the different tools, we have designed and implemented a component-based software, EULIA, that facilitates operations on the linguistic annotations.

Keywords: linguistic annotations, NLP software engineering, stand-off annotation

1 Introduction

In this paper we present a framework for creating, browsing and editing linguistic annotations generated by a set of different linguistic processing tools[1](Artola *et al.* 00).

The objective is to establish a flexible and extensible infrastructure for consulting, visualizing, and modifying annotations generated by existing linguistic tools, following a coherent and general representation scheme (Artola *et al.* 02).

The main goal of this proposal is to set up a well-formalized basis for the exchange of linguistic information among tools. We use TEI-P4 conformant (http://www.tei-c.org/P4X/DTD/) typed feature structures as a representation schema for linguistic analyses.

We have identified the consistent underlying data model which captures the structure and relations contained in the information to be manipulated. This data model is represented by classes that are encapsulated in several library modules, following the object-oriented paradigm.

[1]URL: http://ixa.si.ehu.es

Besides, we have also implemented EULIA, an extensible, component-based software architecture to integrate language engineering applications. EULIA is a user-oriented linguistic data manager, with an intuitive and easy-to-use GUI that offers help in data browsing, manual disambiguation and annotation tasks.

The rest of the paper is organized as follows. In section 2 we present some related work. Section 3 will be dedicated to explain the annotation framework proposed; that is, the representation scheme used for the linguistic information obtained from the different tools. In section 4 we explain the information flow among the different linguistic processors integrated so far. Section 5 presents LibiXaML, the program library which deals with the different types of linguistic information, i.e., with what we call the "annotation web". In section 6, the library-oriented approach we use to store information is presented. Section 7 describes EULIA, an application implemented for facilitating the work with the annotation web. Finally, section 8 presents conclusions and future work.

2 Related work

There is a general trend for establishing standards for effective language resource management (ISO/TC 37/TC 4 (Ide & Romary 04)), the main objective of which is to provide a framework for language resource development and use. A key issue in software development in NLP processes is the definition of a framework for linguistic knowledge representation. Such a framework has to satisfy needs entailed by the different tools and has to be general enough (Basili *et al.* 98). It is not trivial to adopt a formalism to represent this information and different approaches have been considered for this task. For example, ALEP (Advanced Language Engineering Platform) (Simkins 94) can be considered the first integrating environment for NLP design, where all the components (linguistic information, processing modules and resources) are homogeneously described using the ALEP User Language (AUL) based on a DAG formalism. Perhaps the most influential system in the area is GATE (Cunningham *et al.* 96; Bontcheva *et al.* 04; Neff *et al.* 04) which provides a software infrastructure on which NLP applications may be combined into larger application systems. Following this tendency, ATLAS and

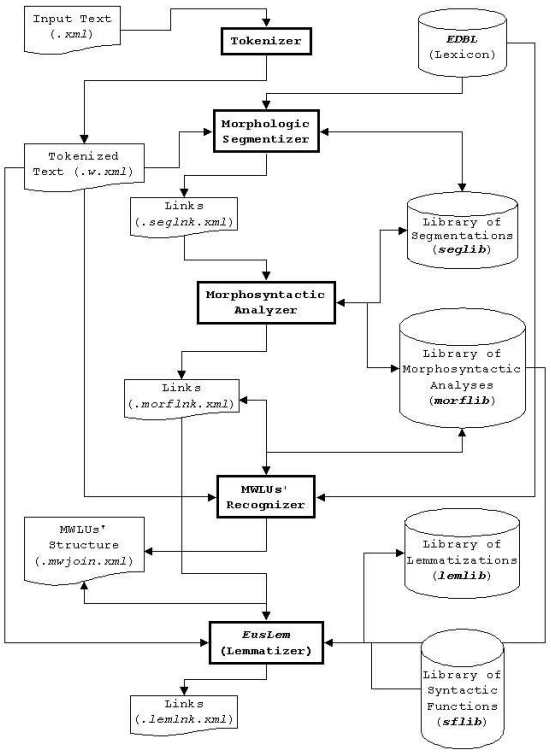

Figure 1: The multi-document annotation web (1)

MAIA (Bird *et al.* 00; Laprun *et al.* 02) provide an architecture targeted at facilitating the development of lingistic annotation applications. In Talent (Neff *et al.* 04), the authors present a pipeline architecture allowing for rapid prototyping and application development. The UIMA model (Ferrucci & Lally 04) permits the implementation of middleware frameworks that allow component-based infrastructure for enabling the rapid combination of linguistic technologies.

The annotation framework presented in this paper follows the stand-off markup approach and it has been inspired on TEI-P4 guidelines (Sperberg-McQueen & Burnard 02) to represent linguistic information obtained by a wide range of linguistic tools. The reason for taking this approach is that our representation requirements are not completely fulfilled by the annotation schemes proposed in the systems mentioned before. For instance, the TIPSTER architecture [Grishman 97] used in GATE version 1 exhibits problems when encoding some linguistic structures, as those referred to non-continuous multiword lexical units. The ATLAS system, based on the so-called Directed Annotation Graphs (DAG) for annotation purposes, exhibits the same restrictions. GATE version 2 (Cunningham *et al.* 02) tried to solve this problem combining TIPSTER and DAG, but the solution they propose makes the annotation of some simple, non-continuous features complex and non-intuitive.

Basque is an agglutinative language and the morphological information we want to attach to every word-form obliges us to use a rich model to represent it. The models used by these well-known systems don't fullfil this requirement. Our stand-off markup annotation system can represent any kind of linguistic information or structure. Following the TEI guidelines, we can deal with any kind of linguistic annotation by means of few elements such us anchors, joins, links and feature structures.

3 The annotation framework

Two main features characterize our annotation framework:

1. The variety of anchors to which the linguistic information can be attached ranges from single tokens, continous and discontinous multi-token lexical units, and different kinds of spans up to even particular word interpretations.

2. The richness and complexity of the linguistic information we need to represent. For example, in morphological analysis, we want to describe phenomena such as intra-word ellipsis or the inner structure of derivatives and compounds

In our case, within this framework of stand-off linguistic annotation, the output of each analysis tool may be seen as composed of several XML documents: *the annotation web*. Figure 1 and Figure 2 show the currently implemented document model including the representation schemes used in tokenization, segmentation, morphosyntactic analysis, multiword recognition, lemmatization/disambiguation, shallow syntax

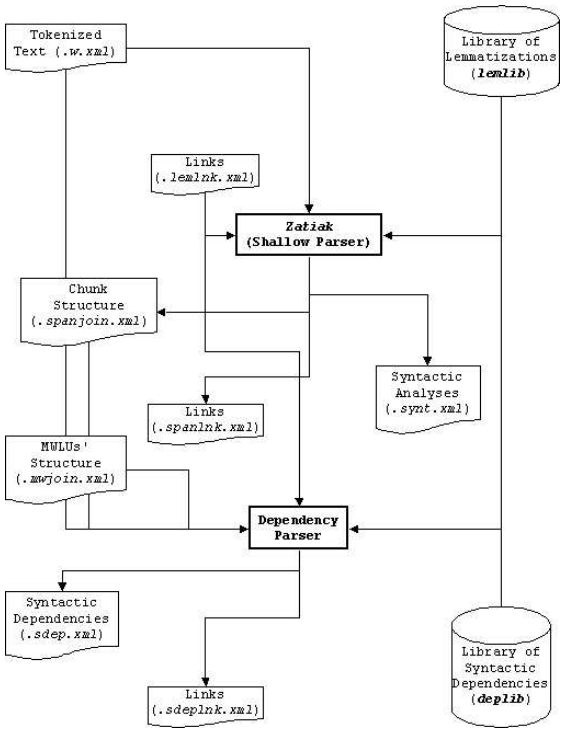

Figure 2: The multi-document annotation web (2)

and dependency-based analysis. This model fulfils the general requirements proposed in the standards (Ide & Romary 04), as in (Bird *et al.* 00; Schäfer 03):

- It provides a way to represent different types of linguistic information, ranging from the general to the fine-grained one where partial results and ambiguities can be easily represented.
- It uses feature structures as a general data model, thus providing a formal semantics and a well known logical operation set over the linguistic information represented by them.
- A general abstract model has been identified over the particular linguistic processors. Therefore, NLP applications are able to import/export the information they need in a unified way.
- The representation model doesn't dependend on any linguistic theory nor any particular processing software.

3.1 The annotation web

As said above, linguistic information is attached to the analyzed text and represented as a set of XML documents that constitute the annotation web. Looking at the characteristics of the documents to be generated, we have identified different groups and types of documents. Next, we will present all of them indicating the elements defined and the corresponding class used for their representation in our model.

- Text anchors: text elements found in the input text.
  - Single-word tokens recognized by the tokenizer. They are tagged with the XML <w> element, and represented in our model by the W class.
  - Multiword lexical units: the collection of "multiword tokens" identified in the input. The MWSTRUCT class represents the constituents of a multiword unit, which is encoded by means of a <join> element that gathers the individual constituents of the unit.
  - The structure of syntactic chunks recognized in the text: the collection of "spans" identified in the input. The SPANSTRUCT class represents the constituents of a chunk that are also tagged by means of <join> elements.
- Analysis collections: collections of linguistic analyses obtained by the different tools. Due to the complexity of the information to be represented we decided to use feature structures (FS) as a general data structure. The use of feature structures quickly spread to other domains within linguistics since Jacobson (Jacobson 49) first used them for the representation of phonemes. Feature structures serve as a general-purpose linguistic metalanguage; this reason led us to use them as the basis of our encoding. The feature structures we use fulfill the TEI's guidelines for typed FSs, and the schema of all the inputs/outputs in the tool pipeline has been thoroughly described by means

of Relax NG Schemas.

- Links between anchors and their corresponding analyses, tagged by means of <link> elements. They are represented by the LINK class

The multi-document annotation web gives, as pointed out in (Ide & Véronis 95; Ide & Romary 04), more independence and flexibility to the different processes, and greater facilities for their integration. In figure 3 we will show an example which illustrates how the multi-document annotation web looks like once the lemmatization process is carried out.

4 The I/O stream between programs

There are many linguistic tools integrated so far. Figure 1 and 2 illustrate the integration of the lexical database (Aldezabal *et al.* 01) and the rest of the tools, emphasizing that the communication among the different processes is made by means of XML documents. Let us describe these processes in sequence:

1. Having an XML-tagged input text file, the tokenizer takes this file and creates, as output, a *.w.xml* file, which contains the list of the tokens and sentences recognized in the input text. The tokenized text is of great importance in the process, in the sense that it intervenes as input for different processes.

2. After the tokenization process, the segmentizer takes as input the tokenized text and the general lexicon issued from the lexical database, and updates the library of segmentation analyses (FSs describing the different morphemic segments found in each word token; one FS per different distinct word-form) producing as well a document (*.seglnk.xml*) containing the links between the tokens in the *.w.xml* file and their corresponding analyses (one or more) in the library. The stand-off framework we follow in annotating the documents allows us to attach easily different analyses to one token.

3. After that, the morphosyntactic treatment module takes as input the output of the segmentation process and updates the library of morphosyntactic analyses *morflib*. It produces a *.morflnk.xml* document containing the links between the tokens in the *.w.xml* file and their corresponding analyses (one or more) in *morflib*.

   The library *morflib* will be later enriched by the MWLUs' treatment module (Alegria *et al.* 04). This module performs the processing of multiword lexical units producing a document that describes, by means of a collection of <join> elements *.mwjoin.xml*, the structure of the MWLUs identified in the text. This module has obviously access to the morphosyntactic analyses and to the *.morflnk.xml* document, into which it will add the links between the *.mwjoin.xml* document and the library.

4. The morphosyntactic analyses and the output of the tokenizer constitute the input of the *Euslem* lemmatizer (Ezeiza *et al.* 98). The lemmatizer updates the library of lemmatizations and produces the *.lemlnk.xml* document that contains the links between the tokens and MWLUs, and their corresponding lemmatization analyses. Besides, it updates the *.mwjoin.xml* document removing the incorrect joins previously included in it.

5. In figure 2, the syntactic process is depicted. The *Zatiak* surface syntax parser (Aduriz *et al.* 04) identifies the chunks in the text (phrases, verb chains and so on) based on the syntactic functions that, following the Constraint Grammar formalism (Karlsson *et al.* 95), the lemmatizer has associated to each word of the text. In this process a named-entity recognizer is also included. This process produces three documents: a *.spanlnk.xml* document that describes which tokens and MWLUs belong to each chunk in the text; a *.synt.xml* document that contains syntactic features associated to each chunk; and a *.spanjoin.xml* document containing the links between the chunks and the *.synt.xml* document. Note that the syntactic analyses contained in the *synt.xml* document correspond to a single input text, since, obviously, there is no general library containing syntactic analysis.

6. Finally, a dependency grammar parser establishes the dependencies between the components of the sentence in order to obtain a syntactic tree. It takes as input the library of the different syntactic dependencies *deplib* and obtains an *.sdep.xml* document describing the syntactic dependencies found in the sentences (Aranzabe *et al.* 04) and a *.sdeplnk.xml* document containing the links between the dependencies and the library.

Figure 3 shows a sample of the annotation web, result of the lemmatization. The input-text is at the upper-left part of the drawing. A multiword expression *Hala ere* (Basque for *however*) and a single-word token *ere* (Basque for *also*) have been emphasized to illustrate the relationships established between these items in the text and their corresponding lemmatizations represented by feature structures contained in the document at the upper-right part of the drawing. The document called *tokenized text* contains the results of the tokenization process: the sequence of tokens identified in the text with the indication of the character offsets corresponding to each token in the source. Similarly, a document called *MWLU's structure* contains the results of the MW expressions processing: the sequence of multi-word elements identified

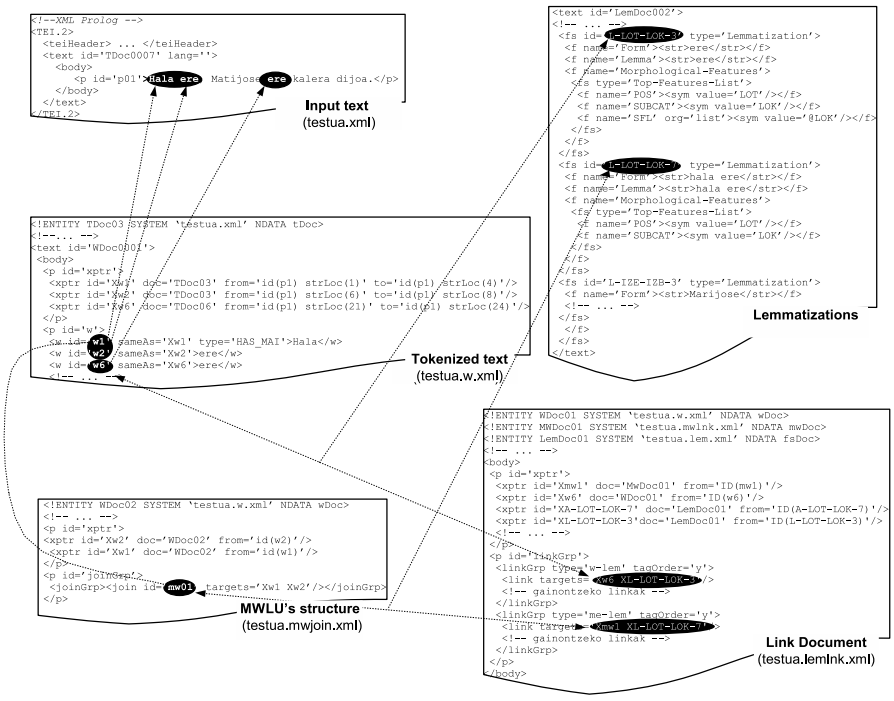

Figure 3: Output of the lemmatizer: a sample of the multi-document annotation web

in the text with the indication of the single tokens belonging to each one of them. Finally, the actual annotations (and the ambiguities, if any) are represented by the link document, that attaches the different items in the source text (single- or multi-word tokens) to their corresponding lemmatizations. TEI external pointers are used to refer to elements not present in the same document.

5 LibiXaML: A program library for dealing with the annotation web

We identified the consistent underlying data model which captures the structure and relations contained in the information to be manipulated. This data model is represented by classes which are encapsulated in several library modules, following the object oriented paradigm. These modules offer the necessary types and operations the different tools need to perform their task when recognizing the input and producing their output. LibiXaML manipulates: features, feature structures, values, XML documents containing linguistic information of different types, document headers, and so on.

The class methods in LibiXaML allow:

- Getting the necessary information from an XML document containing tokens, links, multiword structure joins, FSs, etc.
- Producing with ease the corresponding output according to a well-defined XML description.

The class library has been implemented in C++ and it contains about 100 classes. For the implementation of the different classes and methods we make use of the LT XML system (Thompson *et al.* 97), a tool architecture for XML-based processing of text corpora. The current release of LibiXaML works on Unix and can be soon found at `http://ixa.si.ehu.es/ixa/resources/libixaml`.

6 Storing linguistic information in general FS-Libraries

Considering the huge amount of information obtained in these linguistic processes, it is crucial to get an optimal storage of data in order to provide a fast answer when retrieving and searching this information. We are experimenting two ways of doing things:

- Document-oriented approach: the segmentations, morphosyntactic analyses and lemmatizations (FSs) obtained by the different analysis tools applied on a given document constitute FS collections which are stored in files specifically attached to that document.
- Library-oriented approach: the segmentations, morphosyntactic analyses and lemmatizations (FSs) obtained by the different analysis tools applied on a given document are added to general FS collections stored in big FS libraries.

The second approach saves lots of disk space and speeds up the analysis procedures because the analysis of most word forms must not be repeated since their results will be already stored in the library. So, performing the analysis is just a matter of retrieving the

corresponding analysis identifiers in the library and establishing a link to them.

Using the library-oriented technique to store information requires a more powerful indexing scheme, which will avoid, most of the time, the access to the actual XML FS library.

In order to test our annotation framework on a running environment, we have tokenized, segmentized and morphologically analyzed a text corpus containing 426,205 tokens (71,893 of them are the punctuation marks). The annotation web issued from morphological analysis has been stored, according to the library-based approach, in the following manner:

- feature structures that represent the morphological analyses corresponding to the word forms in the corpus (one for each different word form) have been loaded on an XML-native database (Berkeley DBXML);
- links between text elements and their corresponding analyses have been stored in a relational database (Berkeley DB) for faster retrieval;
- tokenization results and the original text are left in the file system.

A query prototype has been developed on this architecture and some experiments have been carried out on it. For now, this prototype provides us with a quite basic functionality, allowing to pose complex XPath expressions as queries. The XPath expressions are evaluated against the morphological analyses in the XML database that has been adequately indexed, returning as result the identifiers of the feature structures that meet the constraints expressed by the query; next, these identifiers are searched in the relational database containing the links in order to get the identifiers of the corresponding tokens, which are then retrieved on the original text to get their contexts. The final result is that, for example, to get a concordance (KWIC) that contains the words whose morphological analyses meet the constraints in the query along with their contexts takes around one second in a SUN workstation.

7 EULIA: An application to create, browse and disambiguate linguistic annotation based on the annotation web

In order to work on the annotation framework here explained, we have developed EULIA an environment that implements an extensible, component-based software architecture to integrate natural language engineering applications and to exploit the data created by these applications. The main functions of EULIA are the following ones:

- search, queries and analysis of results.
- submit a text to be analyzed.
- consultation and browsing of the linguistic annotation attached to texts.
- manual disambiguation of analysis results.
- manual annotation facilities and suitable encoding for new linguistic information.
- personalization of users.

Regarding the interface, the main window is divided into two parts: a Multi-Document Interface (MDI) panel where linguistic information is shown in an understandable way. The interface provides hypertextual facilities, showing the linguistic information associated to items selected on the left part. The environment is designed as a tool for general users and linguists.

8 Conclusion and future work

In this paper we present a framework for dealing with language annotations.

Our proposal provides a flexible and extensible infrastructure for consulting, visualizing, and modifying annotations generated by existing linguistic tools. In this framework, the fact that different analysis sets (segmentations, complete morphosyntactic analyses, lemmatization results, and so on) linked to text anchors are stored in analysis libraries in a stand-off fashion implies a reduction in time and space resources. Regarding the physical storage of the annotation information, we have already implemented the document-based storage approach and are now refining the library-based approach previously explained because we think that the use of XML native databases should be a good solution for fast retrieval and searching on these huge analysis libraries. So, we are planning to move progressively to this library-based approach. The work done so far confirms the scalability of our approach.

Very few studies have used the stand-off markup based on TEI-P4 guidelines. From our point of view, the TEI-P4 approach gives us the expresiveness required by the complexity of the linguistic information we want to represent both when establishing diverse kinds of anchors to which attach information, and when defining specialized FS types for this information.

We have designed and implemented LibXaML, a component-based library that represents the different types of information to be manipulated.

Morever, EULIA, an extensible, component-based software architecture to integrate natural language engineering applications facilitates the work on these annotations offering help in data browsing, manual disambiguation and annotation tasks.

9 Acknowledgements

This research was partially funded by the Basque Government and University (HIZKING21 project and 9/UPV00141.226-14601/2002)

References

(Aduriz *et al.* 04) Itziar Aduriz, Maxux Aranzabe, Jose Mari Arriola, Arantza Díaz de Ilarraza, Koldo Gojenola, Maite Oronoz, and Larraitz Uria. *Computational Linguistics and Intelligent Text Processing*, chapter A Cascaded Syntactic Analyser for Basque, pages 124–135. 2945 LNCS Series. Springer Verlag, 2004.

(Aldezabal *et al.* 01) Izaskun Aldezabal, Olatz Ansa, Bertol Arrieta, Xabier Artola, Aitzol Ezeiza, Gregorio Hernández, and Mikel Lersundi. EDBL: a general lexical basis for the automatic processing of Basque. In *IRCS Workshop on linguistic databases.*, Philadelphia. USA, 2001.

(Alegria *et al.* 04) Iñaki Alegria, Olatz Ansa, Xabier Artola, Nerea Ezeiza, Koldo Gojenola, and Ruben Urizar. Representation and Treatment of Multiword Expressions in Basque. In *ACL workshop on Multiword Expressions*, Barcelona, 2004.

(Aranzabe *et al.* 04) Maxux Aranzabe, Jose Mari Arriola, and Arantza Díaz de Ilarraza. Towards a dependency parser for Basque. In *Proc. of International Conference on Computational Linguistics. COLING'2004*, Geneva, 2004.

(Artola *et al.* 00) Xabier Artola, Arantza Díaz de Ilarraza, Nerea Ezeiza, Koldo Gojenola, Aitor Maritxalar, and Aitor Soroa. A proposal for the integration of NLP tools using SGML-Tagged documents. In *Proc. of the Second Int. Conf. on Language Resources and Evaluation*, Athens (Greece), 2000.

(Artola *et al.* 02) Xabier Artola, Arantza Díaz de Ilarraza, Nerea Ezeiza, Koldo Gojenola, Gregorio Hernández, and Aitor Soroa. A class library for the integration of NLP tools: Definition and implementation of an abstract data type collection for the manipulation of SGML documents in a context of stand-off linguistic annotation. In *Proc. of the Third Int. Conf. on Language Resources and Evaluation*, Las Palmas (Spain), 2002.

(Basili *et al.* 98) Roberto Basili, Massimo Di Nanni, and Maria Teresa Pazienza. Engineering of IE Systems: An Object-oriented approach. In Maria Terese Pazienza, editor, *Information Extraction: Towards scalable, Adaptable Systems*, volume 1714 of *Lecture Notes in Artificial Intelligence*, pages 134–164. Springer-Verlag, 1998.

(Bird *et al.* 00) Steven Bird, David Day, John Garofolo, Henderson Henderson, Christophe Laprun, and Mark Liberman. ATLAS: A flexible and extensible architecture for linguistic annotation. In *Proc. of the Second International Conference on Language Resources and Evaluation*, pages 1699–1706, Paris (France), 2000.

(Bontcheva *et al.* 04) Kalina Bontcheva, Valentin Tablan, Diana Maynard, and Hamish Cunningham. Evolving GATE to meet new challenges in language engineering. *Natural Language Engineering*, 10(3-4):349–373, 2004.

(Cunningham *et al.* 96) Hamish Cunningham, Yorick Wilks, and Robert J. Gaizauskas. GATE: a General Architecture for Text Engineering. In *Proceedings of the 16th conference on Computational linguistics*, pages 1057–1060. Association for Computational Linguistics, 1996.

(Cunningham *et al.* 02) H. Cunningham, D. Maynard, K. Bontcheva, V. Tablan, and C. Ursu. *The GATE User Guide.* 2002. http://gate.ac.uk/.

(Ezeiza *et al.* 98) Nerea Ezeiza, Itziar Aduriz, Iñaki Alegria, Jose Mari Arriola, and Ruben Urizar. Combining Stochastic and Rule-based Methods for Disambiguation in Agglutinative Languages. In *Proc. of COLING-ACL'98*, pages 10–14, Montreal (Canada), 1998.

(Ferrucci & Lally 04) David Ferrucci and Adam Lally. UIMA: an architectural approach to unstructured information processing in the corporate research environment. *Natural Language Engineering*, 10(3/4):327–348, 2004.

(Ide & Romary 04) Nancy Ide and Laurent Romary. International standard for a linguistic annotation framework. *Natural Language Engineering*, 10(3-4):211–225, 2004.

(Ide & Véronis 95) Nancy Ide and Jean Véronis, editors. *Text Encoding Initiative. Background and Context.* Kluwer Academic Pub, 1995.

(Jacobson 49) Roman Jacobson. The identification of phonemic entities. *Travaux du Cercle Linguistique de Copenhague*, 5:205–213, 1949.

(Karlsson *et al.* 95) Fred Karlsson, Atro Voutilainen, Juha Heikkilä, and Arto Anttila, editors. *Constraint Grammar: A Language-independent System for Parsing Unrestricted Text*, volume 4 of *Natural Language Processing.* Natural Language Processing, Mouton de Gruyter, Berlin and New York, 1995.

(Laprun *et al.* 02) Cristophe Laprun, Jonathan. Fiscus, John. Garofolo, and Silvai. Pajot. A practical introduction to ATLAS. In *Proceedings of the Third International Conference on Language Resources and Evaluation*, 2002.

(Neff *et al.* 04) Mary S. Neff, Roy J. Byrd, and Branmir K. Bougaraev. The Talent system: TEXTRACT architecture and data model. *Natural Language Engineering*, 10(3-4):307–326, 2004.

(Schäfer 03) Ulrich Schäfer. WHAT: An XSLT-based infrastructure for the integration of natural language processing components. In *Proceedings of the Workshop on the Software Engineering and Architecture of Language Technology Systems (SEALTS), HLT-NAACL03*, Edmonton (Canada), 2003.

(Simkins 94) N. K. Simkins. An Open Architecture for Language Engineering. In *First CEC Language Engineering Convention*, Paris (France), 1994.

(Sperberg-McQueen & Burnard 02) C. M. Sperberg-McQueen and L. Burnard, editors. *TEI P4: Guidelines for Electronic Text Encoding and Interchange.* Oxford, 4 edition, 2002.

(Thompson *et al.* 97) H.S. Thompson, R.Tobin, D. Mckelvie, and C. Brew. *LT XML Software API and toolkit for XML processing.* 1997. http://www.ltg.ed.ac.uk/software/xml/index.html.