Procesamiento del Lenguaje Natural, nº39 (2007), pp. 157-164 recibido 17-05-2007; aceptado 22-06-2007

Specification of a general linguistic annotation framework and its use in a real context

Xabier Artola, Arantza Díaz de Ilarraza, Aitor Sologaistoa, Aitor Soroa
IXA Taldea
Euskal Herriko Unibertsitatea (UPV/EHU)
xabier.artola@ehu.es

Resumen: AWA es una arquitectura general para representar información lingüística producida por procesadores lingüísticos. Nuestro objetivo es definir un esquema de representación coherente y flexible que sea la base del intercambio de información entre herramientas lingüísticas de cualquier tipo. Los análisis linguísticos se representan por medio de estructuras de rasgos según las directrices de TEI-P4. Estas estructuras y su relación con los demás elementos que componen el análisis forman parte de un modelo de datos diseñado bajo el paradigma de orientación a objetos. AWA se encarga de la representación de la información dentro de una arquitectura más amplia para gestionar todo el proceso de análisis de un corpus. Como ejemplo de la utilidad del modelo presentado explicaremos cómo se ha aplicado dicho modelo en el procesamiento de dos corpus.
Palabras clave: Modelo de anotación, arquitectura para la integración, TEI-P4

Abstract: In this paper we present AWA, a general architecture for representing the linguistic information produced by diverse linguistic processors. Our aim is to establish a coherent and flexible representation scheme that will be the basis for the exchange of information. We use TEI-P4 conformant feature structures as a representation schema for linguistic analyses. A consistent underlying data model, which captures the structure and relations contained in the information to be manipulated, has been identified and implemented by a set of classes following the object-oriented paradigm. As an example of the usefulness of the model, we will show the usage of the framework in a real context: two corpora have been annotated by means of an application which aim is to exploit and manipulate the data created by the linguistic processors developed so far.
Keywords: Annotation model, integration architecture, TEI-P4

1 Introduction

In this paper we present AWA (Annotation Web Architecture), which forms part of LPAF, a multi-layered Language Processing and Annotation Framework. LPAF is a general framework for the management and the integration of NLP components and resources. AWA defines a data representation schema which aim is to facilitate the communication among linguistic processors in a variety of NLP applications. The key design criteria we have taken into account when designing AWA are oriented to make possible the description of different phenomena in an homogeneous way.

The objective of AWA is to establish a coherent and flexible representation scheme that will be the basis for the exchange of information. We use TEI-P4 conformant feature structures[1] to represent linguistic analyses. We also have identified a 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 that are encapsulated in several library modules (LibiXaML), following the object-oriented paradigm(Artola et al., 2005). The modules offer the necessary types and operations to manipulate the linguistic information according to the model. The class library has been implemented in C++ and contains about 100 classes. For the implementation of the different classes and methods we make use of the Libxml2[2] library.

[1] http://www.tei-c.org/P4X/DTD/
[2] http://xmlsoft.org/

ISSN: 1135-5948

The current release of LibiXaML works on Unix flavours as well as on Windows architectures.

As an example of the usefulness of the model we will show the usage of the framework in a real context. Two corpora have been tagged by means of an on-line application, called EULIA, which aim is to exploit and manipulate the data created by the linguistic processors developed so far and integrated in a pipeline architecture. EULIA (Artola et al., 2004) offers help in data browsing, manual disambiguation, and annotation tasks by means of an intuitive and easy-to-use graphic user interface.

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 proposed annotation architecture. In section 4 we describe the use of feature structures for representing linguistic information. Section 5 shows the use of the framework in two real contexts: the annotation of EPEC (Reference Corpus for the Processing of Basque) and ztC (Science and Technology Corpus) (Areta et al., 2006), and EULIA, an application implemented for facilitating the work with the so-called annotation web. Finally, section 6 is dedicated to present some 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 and Romary, 2004)), the main objective of which is to provide a framework for language resource development and use. Besides, there is much work dealing with the use of XML-based technologies for annotating linguistic information. ATLAS (Bird et al., 2000), LT-TTT (Thompson et al., 1997) and WHAT are some of the projects where stand-off annotation is used in order to deal efficiently with the combination of multiple overlapping hierarchies that appear as a consequence of the multidimensional nature of linguistic information. LT-TTT (Thompson et al., 1997) is a general library developed within an XML processing paradigm whereby tools are combined together in a pipeline allowing to add, modify and remove pieces of annotation. It provides linguistic components that operate over XML documents and permit the development of a broad range of NLP applications which share the annotated information. In ATLAS (Bird et al., 2000) the authors use XML technology as a format for the interchange of annotated information between linguistic applications (AIF). In a first version, ATLAS was fully based in a particular formalism for annotation, called Annotation Graphs (AGs). However, they extended the architecture in order to adopt an upper level of abstraction and provide an ontology, where the conceptual model can be defined. For this reason MAIA (Meta Annotation Information for Atlas) is defined (Laprun et al., 2002)). Although the ontology model is described in XML documents, no XML technology is used to semantically validate the information. Finally, in the WHAT project (Schäfer, 2003), the authors present an XSLT-based Whiteboard Annotation Transformer, an integration facility for integrating deep and shallow NLP components. They rely on XSLT technology for transforming shallow and deep annotations in an integrated architecture built on top of a standard XSL transformation engine. Linguistic applications communicate with the components through programming interfaces. These APIs are not isomorphic to the XML mark-up they are based on, but they define classes in a hierarchical way. Among other types of formalisms they use typed feature structures for encoding deep annotations, although the correctness of these feature structures is not validated with XML tools.

Apart from the annotation infrastructure, several systems go further and define frameworks for rapid prototyping of linguistic applications that share the same data model (annotations) at different levels. GATE (Cunningham, Wilks, and Gaizauskas, 1996; Bontcheva et al., 2004), TALENT (Neff, Byrd, and Bougaraev, 2004), ATLAS and MAIA (Bird et al., 2000; Laprun et al., 2002), and UIMA (Ferrucci and Lally, 2004)) are some of these systems.

The annotation architecture presented in this paper follows the stand-off markup approach and it has been inspired on the TEI-P4 guidelines (Sperberg-McQueen and Burnard, 2002) to represent linguistic information obtained by a wide range of linguistic tools.

One reason for taking this approach is that our representation requirements, together with the characteristics of the lan-

guage (Basque) we are dealing with, are not completely fulfilled by the annotation schemes proposed in the systems mentioned before. Basque being an agglutinative and free-order language, the complexity of the morphological information attached to linguistic elements (word-forms, morphemes, multiword expressions, etc.) as well as the need to represent discontinuous linguistic units, obliges us to use a rich representation model.

3 The annotation architecture in a language processing framework

In this section, the general annotation web architecture (AWA) is described from an abstract point of view, and situated within LPAF.

3.1 Language Processing and Annotation Framework

Figure 1 depicts the main components of LPAF. The framework has been organized in different layers.

The bottom layer defines the basic infrastructure shared by any LPAF component. In this layer we can find:

- The Annotation Web Architecture (AWA), including a set of class libraries which offer the necessary types and operations to manipulate the objects of the linguistic information model (Artola et al., 2005).
- The Linguistic Processing Infrastructure (LPI), which includes the set of classes needed to combine linguistic processes. It is the result of the characterization of the way the linguistic processes interact with each other.

The former will be thoroughly explained in this paper.

The middle layer is formed by the LPAF public services, which constitute the basic resources for defining new linguistic applications. LPAF services perform concrete and well-defined tasks necessary for defining complex linguistic applications such as Q/A systems, environments for manual annotation of corpora at different levels, etc.

On the top layer we can find final user applications. EULIA and the ztC Query System [3] are two examples of this type of applications, and will be explained throughout this paper.

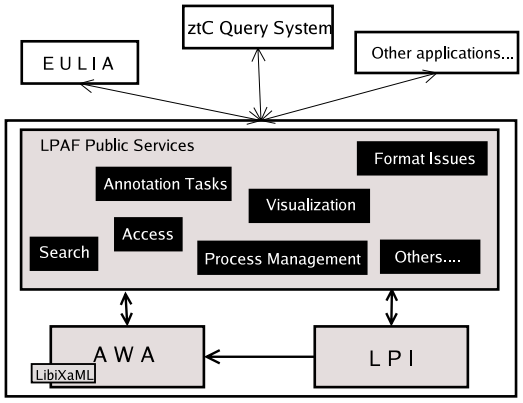

Figure 1: The Language Processing and Annotation Framework

3.2 Annotation Web Architecture

The Annotation Web Architecture has been designed in a way general enough to be used in the annotation tasks of a very broad range of linguistic processing tools. Issues such as the representation of ambiguity or the attachment of linguistic information to units formed by discontinuous constituents have been taken into account in the annotation model.

An abstract view of this annotation architecture is represented in Figure 2. When a text unit undergoes a series of language processing steps, a corpus unit is created. Together with the raw text, this corpus unit includes the linguistic annotations resulting from each of these processing steps. So, each one of these annotations (`LinguisticAnnotation` class) represents, for instance, the set of annotations produced by a lemmatization process or the annotations produced by a dependency-based syntactic parser. Dependencies among different linguistic annotations belonging to the same processing chain are presented by the `dependsOn` association link in the diagram.

The model follows a stand-off annotation strategy: anchors set on the corpus (`Anchor` class) are attached to the corresponding linguistic information (`LingInfo` class) by means of "links" (`AnnotationItem` class). An annotation item always *refers to* one anchor and *has associated* a single fea-

[3] http://www.ztcorpusa.net

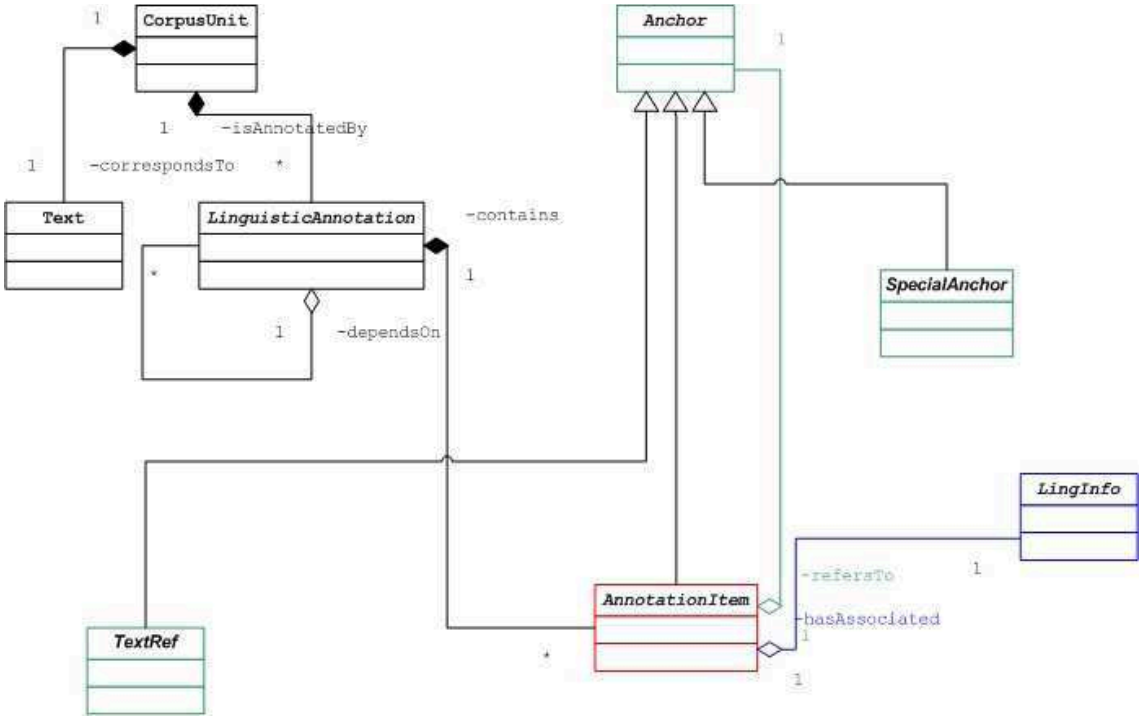

Figure 2: The annotation architecture

ture structure containing linguistic information. Any annotation item can become an anchor in a subsequent annotation operation. As a result of each processing step (tokenization, morphological segmentation or analysis, lemmatization, syntactic parsing, etc.), what we call a "linguistic annotation" consisting of a web of interlinked XML documents is generated.

The model is physically represented by three different types of XML documents: anchor documents, link documents (annotation items) and documents containing linguistic information. Let us show now each one of these in more detail:

- **Anchors**: these elements can go from physical elements found in the input corpus (textual references, represented by the `TextRef` class), such as typical character offset expressions or XPointer expressions pointing to specific points or ranges within an XML document, up to annotation items resulting from previous annotation processes; in particular, morphemes and single- or multiword tokens, word spans, etc., or even "linguistic interpretations" of this kind of elements can be taken as anchors of linguistic annotations. We have found that, in many cases, physical text elements are not adequate as annotation anchors, and linguistic interpretations issued from previous analysis steps (lemmatization and syntactic function combinations, or phrasal chunks to which only some of the interpretations a word can have belong) have to be used as anchors in subsequent processing steps.

  Textual anchors are set mainly as a result of tokenization and of the identification of multiword expressions. On the other hand, interpretational anchors are annotation items or else special anchors (anchors specifically created as "elements" to which attach linguistic information); in this case, they are expressed by XML elements which act as a join of several identifiers representing interpretations issued from previous processes. As examples of special anchors we can mention word sequences, chunks, etc.

  Structural ambiguity is represented by overlapping anchors, i.e., when annotations refer to anchors which overlap.

- **Annotation items (links)**: these constitute the actual annotations resulting from a linguistic analysis process. Each link ties a single linguistic interpretation

to an anchor. Interpretation ambiguity is represented by several links attached to the same anchor, and so disambiguation consists in simply marking one of these links as correct while discarding the rest.

- **Linguistic information**: typed feature structures are used to represent the different types of linguistic information resulting from the analysis processes. In some cases, such as in morphological segmentation or lemmatization, the linguistic content corresponds to word forms (more specifically, *token annotation items*), and therefore huge common libraries containing these contents (feature structures) are used, allowing us to save processing time (and storage room) as previously analyzed word forms need not be analyzed again and again when occurring in new texts.

This data model captures the structure and relations contained in the information to be manipulated, and is represented by classes which are encapsulated in several library modules. These classes offer the necessary operations or methods the different tools need to perform their tasks when recognizing the input and producing their output.

4 *Representing linguistic information: feature structures and Relax NG schemas*

This section is devoted to explain in more detail the use of feature structures in our model, their advantages, features, the representation of meta-information, and the exploitation of schemas in different tasks, such as information retrieval or automatic generation of GUIs.

The different types of linguistic information resulting from the analysis processes are represented as typed feature structures. In a multi-dimensional markup environment, typed feature structures are adequate for representing linguistic information because they serve as a general-purpose metalanguage and ensure the extensibility of the model to represent very complex information. Typed feature structures provide us with a formal semantics and a well-known logical operation set over the represented linguistic information.

The feature structures we use fulfill the TEI guidelines for typed FSs, and they are compatible with ISO/TC 37 TC 4 (Ide and Romary, 2004). Furthermore, we have adopted Relax NG as a definition metalanguage for typed feature structures. Relax NG schemas define the legal building blocks of a feature structure type and semantically describe the represented information.

```
<TEI.2>
...
  <p>
    <fs id="fs1" type="morphosyntactic">
     <f name="Form"><str>esnea</str></f>
     <f name="Lemma"><str>esne</str></f>
     <f name="Morphological-Features">
      <fs type="Top-Feature-List">
       <f name="POS"><sym value="NOUN"/></f>
       <f name="SUBCAT"><sym value="COMMON"/></f>
      </fs>
     </f>
     <f name="Components"> ...</f>
    </fs>
  </p>
  <p>
    <fs id="fs2" type="lemmatization">
     <f name="Form"><str>esnea</str></f>
     <f name="Lemma"><str>esne</str></f>
     <f name="POS"><sym value="NOUN"/></f>
     <f name="SUBCAT"><sym value="COMMON"/></f>
    </fs>
  </p>
...
</TEI.2>
```

Figure 3: Typed feature structures

The type of the feature structure is encoded in XML by means of the `type` attribute (see Figure 3). This attribute allows us to understand the meaning of the information described in the feature structure by means of its link with the corresponding Relax NG schema which specifies the content of the feature structure.

Relax NG schemas provide us with a formalism to express the syntax and semantics of XML documents but, unfortunately, they are not capable of interpreting the content of the feature structures represented in the document. Therefore, we have implemented some tools which, based on the Relax NG schema, arrange data and create automatically the appropriate FS that encodes the associated linguistic information to be represented. These tools can be used to build GUIs for editing linguistic annotations adapting the interface to the user's needs in such a way that they only have to specify the type of the information to be treated. Besides, and thanks to these tools, we are able to build general front- and back-end modules for the integration of different linguistic engines in more complex linguistic applications. Specifying the input/output information by

means of these Relax NG schema for linguistic engines, the front-end module will provide the adequate data to each engine and the back-end module will produce the suitable output.

```
<define name="fs.lemma">
  <element name="fs">
    <attribute name="id"><data type="id"/></attribute>
    <attribute name="type">
     <value>lemmatization</value>
    </attribute>
    <ref name="f.Form"/>
    <ref name="f.Lemma"/>
    <ref name="f.Pos-SubCat"/>
  </element>
</define>

<define name="f.Form">
  <element name="f">
    <attribute name="name"><value>Form</value></attribute>
    <element name="str"><value type="string"/></element>
  </element>
</define>

<define name="f.Pos-Subcat">
  <choice>
    <ref name="pos.Noun"/>
    <ref name="pos.Adj"/>
    ...
  <choice>
<define>

<define name="pos.Noun">
  <ref name="f.POS"/>
  <element name="f">
    <attribute name="name">
     <value>SUBCAT</value>
    </attribute>
    <choice>
     <value>COMMON</value>
     <value>PERSON_NAME</value>
     <value>PLACE_NAME</value>
    <choice>
  </element>
</define>
```

Figure 4: RELAX NG schema mixing morphosyntax and lemmatization

Figure 3. shows a fragment of an XML document which mixes up feature structures of two different linguistic levels (morphosyntactic and lemmatization) for the same word-form. These FSs are defined by the partial Relax NG schema shown in Figure 4. The relation between FSs and the schema is established through the `type` attribute (in both figures in bold). Using these relations, our tools can access the corresponding schemas and exploit them.

5 The use of the annotation architecture in a real context

In order to check the validity of the annotation architecture presented here, we have implemented a pipeline workflow which integrates natural language engines going from a tokenizer to a syntactic parser. Two text corpora have been processed through this pipeline with the aid of a tool named EULIA.

5.1 EULIA: an environment for managing annotated corpora

EULIA is a graphical environment which exploits and manipulates the data created by the linguistic processors. Designed to be used by general users and linguists, its implementation is based on a client-server architecture where the client is a Java Applet running on any Java-enabled web browser and the server is a combination of different modules implemented in Java, C++ and Perl.

The linguistic processors integrated so far in the mentioned architecture are:

- A tokenizer that identifies tokens and sentences from the input text.
- A segmentizer, which splits up a word into its constituent morphemes.
- A morphosyntactic analyzer whose goal is to process the morphological information associated with each morpheme obtaining the morphosyntactic information of the word form considered as a unit.
- A recognizer of multiword lexical units, which performs the morphosyntactic analysis of the multiword expressions present in the text.
- A general-purpose tagger/lemmatizer.
- A chunker or shallow syntactic analyzer based on Constraint Grammar.
- A deep syntax analyzer.

EULIA provides different facilities which can be grouped into three main tasks:

- **Query facility**. It visualizes the answers of the user's requests according to a suitable stylesheet (XSLT). These stylesheets can be changed dynamically depending on both the users' choice and the type of answer.
- **Manual disambiguation**. Its goal is to help annotators when identifying the correct analysis and discarding the wrong ones. The incorrect analyses are properly tagged but not removed.
- **Manual annotation**. It consists of assigning to each anchor its corresponding linguistic information. Depending on the annotation type different kinds of information are needed. In order to get these data, EULIA's GUI generates

a suitable form, based on the Relax NG schema, which defines the document's format for that annotation type. Considering that linguistic information is encoded following the annotation architecture, the treatment at different levels of analysis is similar.

5.2 Annotating ztC and EPEC

Let us now explain briefly two real experiences that demonstrate the flexibility and robustness of the model, the architecture, and the environment built. These experiences have been done on two corpora created with different purposes:

- **ztC Corpus (Science and Technology Corpus)** ztC is a 8,000,000 word corpus of standard written Basque about Science and Technology which aim is to be a reference for the use of the language in Science and Technology texts. Part of this corpus (1,600,000) has been automatically annotated and manually disambiguated. The manual disambiguation of the corpus is performed on the output of EUSTAGGER (Aduriz et al., 1996), a general lemmatizer/tagger that obtains for each word-form its lemma, POS, number, declension case, and the associated syntactic functions. In this case, the manual disambiguation and annotation has been restricted to the information about lemma and POS.
- **EPEC Corpus (Reference Corpus for the Processing of Basque)** EPEC is a 300,000 word corpus of standard written Basque with the aim of being a training corpus for the development and improvement of several NLP tools. The first version of this corpus (50,000 words) has already been used for the construction of some tools such as a morphological analyzer, a lemmatizer, or a shallow syntactic analyzer, but now we are in a process of enhancement by annotating manually 250,000 new words. Although EPEC has been manually annotated at different levels, the manual annotation to which we will refer here has been performed on the output of MORPHEUS (Aduriz et al., 2000), a general analyzer that obtains for each word-form its possible morphosyntactic analyses. EULIA presents this information to the linguist who has to choose the correct one and mark it by means of a facility provided by the application. If the analyzer doesn't offer any correct analysis, the annotator has to produce it filling-up a form obtained automatically in a scheme-based way, as explained in section 4. Once the whole corpus is manually annotated and disambiguated at the segmentation level, the annotations are propagated to other levels (morphosyntax, lemmatization, syntax) automatically and revised again by means of the application. Currently, eight annotators are satisfactorily working in parallel using EULIA.

The flexibility EULIA gets by using Relax NG schemas makes possible to visualize the information needed in each process in such a way that the linguist will only focus on the problem of ambiguity referred to the information given.

6 Conclusions and future work

In this paper we have presented AWA, a general architecture for representing the linguistic information produced by linguistic processors. It is integrated into LPAF, a language processing and annotation framework. Based on a common annotation schema, the proposed representation is coherent and flexible, and serves as a basis for exchanging information among a very broad range of linguistic processing tools, going from tokenization to syntactic parsing.

We have described our general annotation model, where any annotation can be used as anchors of subsequent processes. The annotations are stand-off, so that we can deal efficiently with the combination of multiple overlapping hierarchies that appear as a consequence of the multidimensional nature of linguistic information. Based on our experience, the markup annotation model we propose can represent a great variety of linguistic information or structure.

XML is used as an underlying technology for sharing linguistic information. We have also defined RelaxNG schemas to describe the different types of linguistic information the framework is able to work with. Furthermore, we use these schemas to automatically exploit the information encoded as typed feature structures.

We have also presented EULIA, a graphical environment the aim of which is to exploit and manipulate the data created by the linguistic processors. EULIA offers facilities to browse over the annotation architecture, pose queries and perform manual disambiguation/annotation of corpora.

Finally, we have briefly explained two real cases that show the flexibility and robustness of our annotation model as well as the benefits of an environment like EULIA in manual annotation and disambiguation processes.

References

Aduriz, Itziar, Eneko Agirre, Izaskun Aldezabal, Iñaki Alegria, Xabier Arregi, Jose Mari Arriola, Xabier Artola, Koldo Gojenola, Aitor Maritxalar, Kepa Sarasola, and Miriam Urkia. 2000. A Word-grammar based morphological analyzer for agglutinative languages. In *Proc. of International Conference on Computational Linguistics. COLING'2000*, Saarbrücken (Germany).

Aduriz, Itziar, Izaskun Aldezabal, Iñaki Alegria, Xabier Artola, Nerea Ezeiza, and Ruben Urizar. 1996. EUSLEM: A Lemmatiser / Tagger for Basque. In *EURALEX'96, Part 1, 17-26.*, Göteborg.

Areta, Nerea, Antton Gurrutxaga, Igor Leturia, Ziortza Polin, Rafael Saiz, Iñaki Alegria, Xabier Artola, Arantza Díaz de Ilarraza, Nerea Ezeiza, Aitor Sologaistoa, Aitor Soroa, and Andoni Valverde. 2006. Structure, Annotation and Tools in the Basque ZT Corpus. In *LREC 2006*, Genoa (Italy).

Artola, Xabier, Arantza Díaz de Ilarraza, Nerea Ezeiza, Koldo Gojenola, Aitor Sologaistoa, and Aitor Soroa. 2004. EULIA: a graphical web interface for creating, browsing and editing linguistically annotated corpora. In *LREC 2004. Workshop on XbRAC*, Lisbon (Portugal).

Artola, Xabier, Arantza Díaz de Ilarraza, Nerea Ezeiza, Gorka Labaka, Koldo Gojenola, Aitor Sologaistoa, and Aitor Soroa. 2005. A framework for representing and managing linguistic annotations based on typed feature structures. In *RANLP 2005*, Borovets (Bulgaria).

Bird, Steven, David Day, John Garofolo, Henderson Henderson, Christophe Laprun, and Mark Liberman. 2000. 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).

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

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

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

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

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

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

Schäfer, Ulrich. 2003. 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).

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

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