EULIA: a graphical web interface for creating, browsing and editing linguistically annotated corpora

X. Artola, A. Díaz de Ilarraza, N. Ezeiza, K. Gojenola*, A. Sologaistoa and 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 EULIA, a tool which has been designed for dealing with the linguistic annotated corpora generated by a set of different linguistic processing tools. The objective of EULIA is to provide a flexible and extensible environment for creating, consulting, visualizing, and modifying documents generated by existing linguistic tools. The documents used as input and output of the different tools contain TEI-conformant feature structures (FS) coded in XML. The tools integrated until now are a lexical database, a tokenizer, a wide-coverage morphosyntactic analyzer, a general purpose tagger/lemmatizer, and a shallow syntactic analyzer.

1. Introduction

In this paper we present EULIA, a tool which has been designed for dealing with the linguistic annotated corpora generated by a set of different linguistic processing tools[1](Artola *et al.*, 2000). The objective of EULIA is to provide a flexible and extensible environment for consulting, visualizing, and modifying the documents generated by existing linguistic tools, which follow a coherent and general annotation scheme (Artola *et al.*, 2002).

The interface is based on a general document annotation scheme based on XML. XML provides us with a well-formalized basis for the exchange of linguistic information among the different text analysis tools. TEI-P4 conformant (http://www.tei-c.org/P4X/DTD/) feature structures constitute the representation schema for the different documents that convey the information from one linguistic tool to the next one in the analysis chain. So, XML-coded documents are used as input and output of the integrated tools.

XML is a well-defined standard for representing structured documents. Its value is due to the fact that it closes off the option of a proliferation of ad-hoc notations and the associated software needed to read and write them. The most important reason for using XML to encode the I/O streams between programs is that it forces us to formally describe the mark-up used, and that there exists more and more software available to deal with it.

The rest of the paper is organized as follows. Section 2 will be dedicated to explain the representation we have chosen for the linguistic information obtained from the different tools. In section 3 we present the information flow among the different linguistic processors. Section 4 describes the graphical interface with its main design features. Finally, section 5 presents conclusions and future work.

2. The annotation framework

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.*, 1998). It is not trivial to adopt a formalism to represent this information. Different approaches have been considered for this task. For example, ALEP (Advanced Language Engineering Platform) (Simkins , 1994), can be considered the first integrating environment for NLP design. All the components (linguistic information, processing modules and resources) are homogeneously described using the ALEP User Language (AUL) based on a DAG formalism. Others, like GATE (Cunningham *et al.*, 1996), represent textual information by using the notion of textual annotation firstly introduced in the TIPSTER project.

There is a general trend for establishing standards for effective language resource management (ISO/TC 37/TC 4 (Ide *et al.*, 2003)). The main objective is to provide a framework for language resource development and use.

In our case, within a framework of stand-off linguistic annotation, the output of each of the analysis tools may be seen as composed of several XML documents: *the annotation web*. Figure 1 shows the currently implemented document model including tokenization, segmentation, morphosyntactic analysis, multiword recognition and lemmatization/disambiguation. This model fulfils the general requirements proposed in the standards (Ide *et al.*, 2003), as in (Bird *et al.*, 2000; Schäffer , 2003):

- It provides a way to represent different types of linguistic information, ranging from the very general to the very fine-grained one.
- 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.
- Partial results and ambiguities can be easily represented.
- 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 is not dependent on any linguistic theory nor any particular processing software.

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

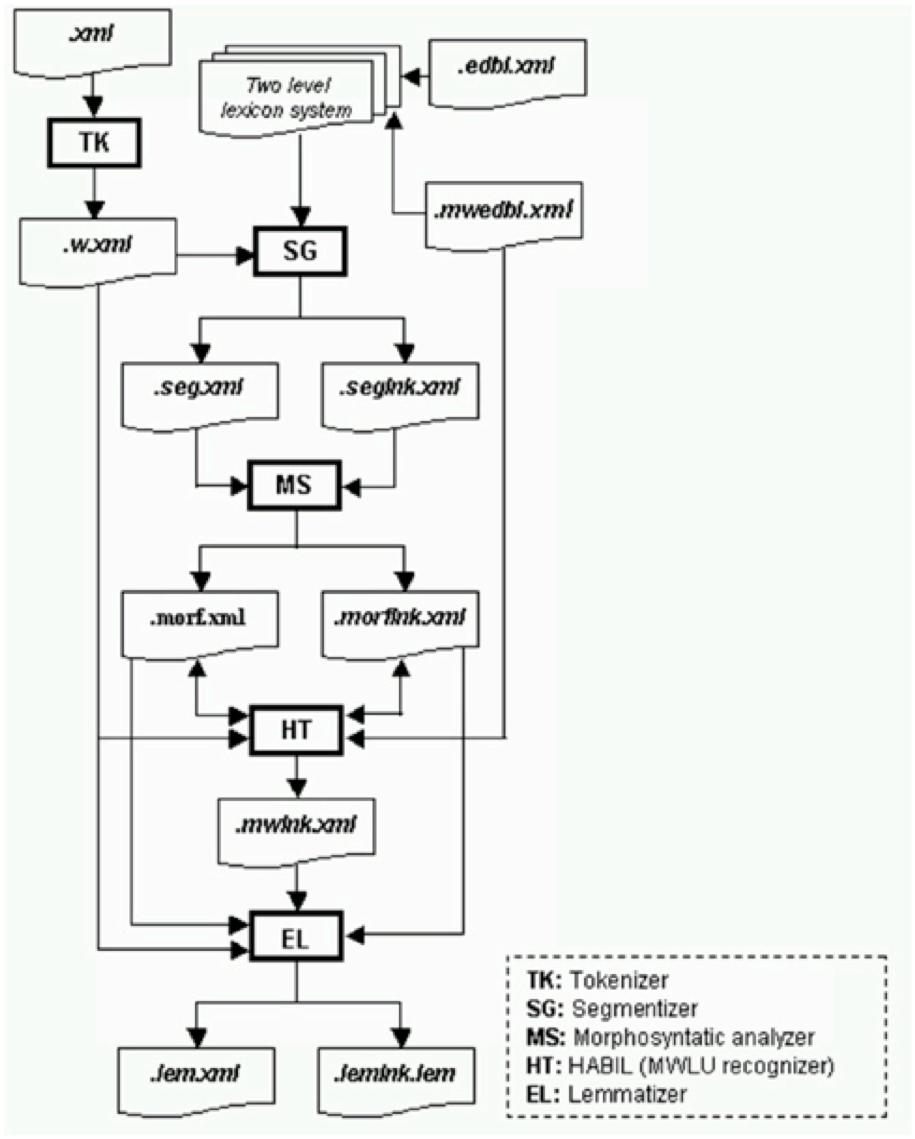


Figure 1: The multi-document annotation web

- As said before, our model relies in the XML mark-up language. XML is a well-defined standard for the representation of structured texts that provides a formal framework for the internal processing. As more and more pieces of software are available for checking the syntactic correctness of the documents, information retrieval, modifications, filtering, and so on, it makes it easy to generate the information in different formats (for processing, printing, screen-displaying, publishing in the web, or translating into other languages).

- Our model guarantees that no different mechanism is used to indicate the same type of information.

We identified the consistent underlying data model which captures the structure and relations contained in the information to be manipulated. These data models are represented by classes which are encapsulated in several library modules. These modules offer the necessary operations the different tools need to perform their task when recognizing the input and producing their output. These functions allow:

- Getting the necessary information from an XML document containing tokens, links, multiword structure links and FSs.

- Producing with ease the corresponding output according to a well-defined XML description.

We have identified different groups and types of documents:

- Text anchors: text elements found in the input.
  - Single-word tokens issued from the tokenizer. They are tagged with the XML <w> element, and represented 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 that are tagged by means of <link> elements. MWSTRUCTL represents lists of MWSTRUCT objects.

  - The structure of the 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 <link> 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 as a general data structure. The use of feature structures quickly spread to other domains within linguistics since Jacobson (1949) 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 in the integrated system are coded following the TEI’s DTD for FSs, and they fulfil the Feature Structure Declarations (FSD) that have been thoroughly described for all the inputs/outputs in the tool pipeline. Following the object oriented paradigm, the following classes have been defined in order to deal with feature structures: FS (feature structure class), FL (list of features of a feature structure), F (feature class), FVL (the list of values of a feature), FVALUE (the value of a feature), and so on. The list of <fs> elements is represented by the class FSL. We distinguish two kinds of collections:
  - Libraries containing the analyses (FSs) corresponding to the text anchors set in the processed texts through the different analysis phases: *seglib*, *morflib*, *lemlib*, *sflib* and *deplib*. They are tagged by means of <fslib> elements.
  - Text-specific documents. Syntactic annotations associated to a particular input text.

- Links between anchors and their corresponding analyses, tagged by means of <link> elements. They are represented by the LINK and LINKL (list of LINK instances) classes.

- Documents: collections of text anchors —single tokens, multiword tokens and spans—, analyses, and links. Several classes to deal with the different kinds of XML documents participating in the annotation web have been defined: list of text elements (WXMLDOC), list of analyses (AXMLDOC), list of links (LNKXMLDOC), list of multiword units (MWXMLDOC), etc.

The multi-document annotation web gives, as pointed out in (Ide and Véronis , 1995; Ide *et al.*, 2003), more independence and flexibility to the different processes, and greater facilities for their integration.

3. The I/O stream between programs

These are the linguistic tools integrated so far:

1. EDBL, a lexical database for Basque, which at the moment contains more than 85,000 entries (Aduriz *et al.*, 1998a)

2. A tokenizer that identifies tokens and sentences from the input text.

3. *Morpheus*, a wide-coverage morphosyntactic analyzer for Basque (Alegria *et al.*, 1996). It attaches to each input word form all its possible interpretations. The result is a set of possible morphosyntactic readings of a word in which each morpheme is associated with its corresponding features in the lexicon: category, subcategory, declension case, number, and definiteness, as well as its syntactic function (Karlsson *et al.*, 1995) and some semantic features. It is composed of several modules such as:
   - A segmentizer, which splits up a word into its constituent morphemes.
   - A morphosyntactic analyzer (Aduriz *et al.*, 2000), whose goal is to group the morphological information associated with each morpheme obtaining the morphosyntactic information of the word form considered as a unit. This is an important step in our analysis process due to the agglutinative character of Basque.
   - A recognizer of multiword lexical units (MWLUs), which performs the morphosyntactic analysis of multiword units present in the text (Aduriz *et al.*, 1996).

4. *EusLem*, a general-purpose tagger/lemmatizer (Ezeiza *et al.*, 1998).

In the future we plan to integrate other tools currently under development, such as a shallow syntactic analyzer based on Constraint Grammar (Karlsson *et al.*, 1995; Aduriz *et al.*, 1998b),

Figure 1 illustrates the integration of the lexical database, the tokenizer, the morphological segmentation, morphosyntactic treatment, treatment of MWLUs, and *EusLem* (lemmatization) emphasizing that the communication among the different processes is made by means of XML documents. Thick line-border rectangles are used to represent processes, which will be described 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 recognized in the input text. The tokenized text is of great importance in the rest of the analysis 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 segmentation analyses library (FSs describing the different morphemic segments found in each word token) producing as well a document (*seg.xml*) containing the links between the tokens in the $w.xml$ file and their corresponding analyses (one or more) in the library. We want to point out that, because of the stand-off strategy followed in annotating the documents, different analyses may be easily attached to one token, thus allowing us to represent ambiguous interpretations.

3. After that, the morphosyntactic treatment module takes as input the output of the segmentation process and updates the library of morphosyntactic analyses. It processes the *seg.xml* document issued in the previous phase producing a *morflnk.xml* document containing the links between the tokens in the $w.xml$ file and their corresponding analyses (one or more) in $morf.xml$. This document will be later enriched by the MWLUs' treatment module. This module performs the processing of multiword lexical units producing an *mw* document that describes, by means of a collection of <link> elements, the structure of the MWLUs identified in the text. This module has obviously access to the morphosyntactic analyses and the *morflnk.xml* document, into which it will add the links between the *mwlnk.xml* document and the library.

4. The morphosyntactic analyses and the output of the tokenizer constitute the input of the lemmatizer. The lemmatizer updates the library of lemmatizations producing two link documents: on the one hand, a *lemlnk.xml* document that contains the links between the tokens and MWLUs, and their corresponding lemmatization analyses. The lemmatizer is also capable of updating the *mwlnk.xml* document if, due to the disambiguation performed, it has to remove some of the incorrect links previously included in it. Figure 2 shows a part of the document collection corresponding to the output of the lemmatizer.

4. EULIA: An application for creation, browsing and disambiguation on the annotation web

In this chapter we describe an extensible, component-based software architecture to integrate natural language engineering applications and to exploit the data created by these applications. The strategy we have explained for the integration of NLP tools is complex, as the linguistic information of different levels is distributed in many documents that must be processed. For any linguistic task it is necessary to coordinate different tools and data sources, and when we add new tools to the production chain, coordination will become more difficult. Therefore, in order to carry out the mentioned strategy, we have defined and implemented EULIA, a web-based interface.

4.1. Main functionalities

EULIA is an environment to coordinate NLP tools and to exploit the data generated by this tools. The NLP tools explained before are integrated in EULIA and new tools are currently being integrated. EULIA has two main goals:

- User-oriented linguistic data manager, with an intuitive and easy-to-use GUI.

- A system to integrate, coordinate and access NLP tools. This task is possible by means of a coordination module and the cooperation of this module with the user interface.

The GUI is a web-based interface which works with XML documents created by the integrated NLP tools. Its main functions are the following ones:

- consultation and browsing of the linguistic annotation attached to texts

- manual disambiguation of analysis results

- manual annotation facilities and suitable codification for new linguistic information

- simple text editor to create new texts

- submit a text to be analyzed in the coordination module

- search, queries and results analysis

- users control and personalization

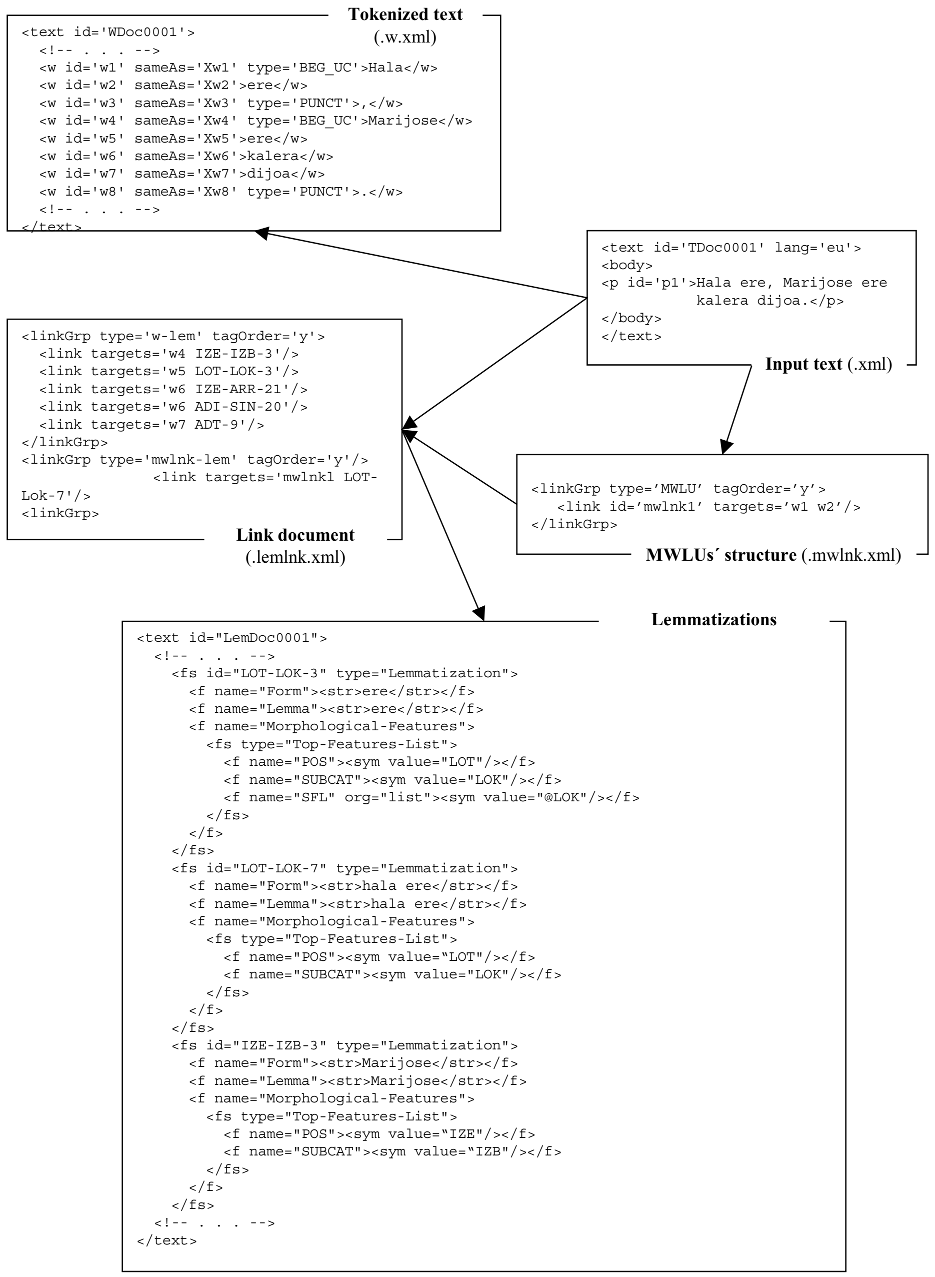
4.2. Architecture and implementation

EULIA's implementation is based on a client-server architecture where the client is a Java Applet accessible by any Java-enabled web browser and the server is a combination of different modules implemented in Java, C++ and Perl (see Figure 4). All modules are designed using an object oriented methodology. As a consequence, EULIA presents a robust design which is easy to extend. The client's goal is to be the intermediary between users and NLP tools. It fulfils users' control and user requests' management. The interface provides different facilities which can be grouped in three main tasks:

- Data browsing: it visualizes the answers of the requests that users make to EULIA. Usually, these requests involve a complex procedure and need the information available in the server to resolve it; that is why the requests are processed by the server. In case, it is necessary to submit an answer to the user, this will be a XML document and will be visualized according to the suitable stylesheet (XSL document). These stylesheets could be changed dynamically depending on both the users' choice and the type of answer.

- Manual disambiguation: because of the integration strategy, disambiguation is an easy task. It consists of eliminating or marking the wrong links among analyses and units (token, multiword, dependencies, etc.) in link documents. EULIA presents a specific interface for this task which is generic for all link documents coded according to TEI guidelines.

- Manual annotation: depending on annotation type, a different kind of information is needed. In order to get these data, EULIA's GUI generates a suitable form, based on the XMLSchema, which defines the document's format for that annotation type. These forms are a HTML document and are generated using XSL documents. Communication between the GUI Applet and the server is established by means of Java Remote Method (RMI), which allows incremental construction of the communication protocol and a natural way to



```
<text id='WDoc0001'>
  <!-- . . . -->
  <w id='w1' sameAs='Xw1' type='BEG_UC'>Hala</w>
  <w id='w2' sameAs='Xw2'>ere</w>
  <w id='w3' sameAs='Xw3' type='PUNCT'>,</w>
  <w id='w4' sameAs='Xw4' type='BEG_UC'>Marijose</w>
  <w id='w5' sameAs='Xw5'>ere</w>
  <w id='w6' sameAs='Xw6'>kalera</w>
  <w id='w7' sameAs='Xw7'>dijoa</w>
  <w id='w8' sameAs='Xw8' type='PUNCT'>.</w>
  <!-- . . . -->
</text>
```

```
<text id='TDoc0001' lang='eu'>
<body>
<p id='p1'>Hala ere, Marijose ere
          kalera dijoa.</p>
</body>
</text>
```

```
<linkGrp type='w-lem' tagOrder='y'>
  <link targets='w4 IZE-IZB-3'/>
  <link targets='w5 LOT-LOK-3'/>
  <link targets='w6 IZE-ARR-21'/>
  <link targets='w6 ADI-SIN-20'/>
  <link targets='w7 ADT-9'/>
</linkGrp>
<linkGrp type='mwlnk-lem' tagOrder='y'/>
               <link targets='mwlnkl LOT-
Lok-7'/>
<linkGrp>
```

```
<linkGrp type='MWLU' tagOrder='y'>
   <link id='mwlnk1' targets='w1 w2'/>
</linkGrp>
```

```
<text id="LemDoc0001">
  <!-- . . . -->
    <fs id="LOT-LOK-3" type="Lemmatization">
      <f name="Form"><str>ere</str></f>
      <f name="Lemma"><str>ere</str></f>
      <f name="Morphological-Features">
        <fs type="Top-Features-List">
          <f name="POS"><sym value="LOT"/></f>
          <f name="SUBCAT"><sym value="LOK"/></f>
          <f name="SFL" org="list"><sym value="@LOK"/></f>
        </fs>
      </f>
    </fs>
    <fs id="LOT-LOK-7" type="Lemmatization">
      <f name="Form"><str>hala ere</str></f>
      <f name="Lemma"><str>hala ere</str></f>
      <f name="Morphological-Features">
        <fs type="Top-Features-List">
          <f name="POS"><sym value="LOT"/></f>
          <f name="SUBCAT"><sym value="LOK"/></f>
        </fs>
      </f>
    </fs>
    <fs id="IZE-IZB-3" type="Lemmatization">
      <f name="Form"><str>Marijose</str></f>
      <f name="Lemma"><str>Marijose</str></f>
      <f name="Morphological-Features">
        <fs type="Top-Features-List">
          <f name="POS"><sym value="IZE"/></f>
          <f name="SUBCAT"><sym value="IZB"/></f>
        </fs>
      </f>
    </fs>
  <!-- . . . -->
</text>
```

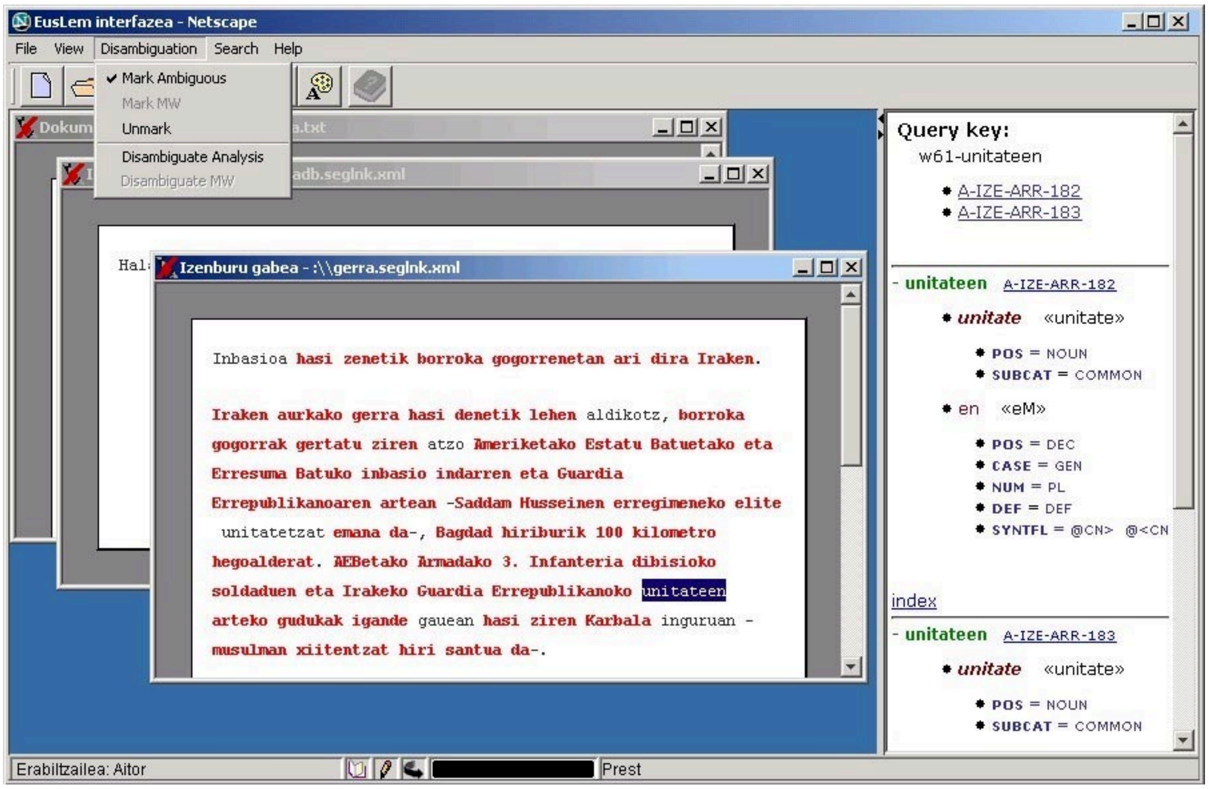
Figure 2: Output of the lemmatizer: a sample of the multi-document annotation web

relate client and server objects. While the client side of the EULIA system consists of an Applet, the server part contains a set of three modules. The first module gives service to clients and it coordinates the integrated NLP tools and stored linguistic analyses.

The second module is a layer between the coordination module and NLP tools. It carries out a generalization of the tools and the analyses.

Finally, the last component is not a module but a set of integrated tools and their outputs.

- Coordination: It coordinates clients' request process and submits the answer in a XML document. In order to answer clients' requests, sometimes it is necessary to generate new linguistic information by the use of integrated tools. Other times, it is enough to search the answer in an existing annotation web. In case it is necessary to generate new information, the system sends the request to the abstraction layer. On the contrary, if the request can be answered from the stored information, we use LibiXaML library to interpret the annotation web and to recover the documents from the abstraction layer. The coordination module has responsibility of managing the set of integrated NLP tools. The final objective of this module is twofold: a) To be the GUI's server and to answer GUI's requests. To solve the requests, this module distributes the tasks among the integrated tools. b) To create a workbench which facilitates the integration of NLP tools and the cooperation among them.



Inbasioa hasi zenetik borroka gogorrenetan ari dira Iraken.

Iraken aurkako gerra hasi denetik lehen aldikotz, borroka gogorrak gertatu ziren atzo Ameriketako Estatu Batuetako eta Erresuma Batuko inbasio indarren eta Guardia Errepublikanoaren artean -Saddam Husseinen erregimeneko elite unitatetzat emana da-, Bagdad hiriburik 100 kilometro hegoalderat. AEBetako Armadako 3. Infanteria dibisioko soldaduen eta Irakeko Guardia Errepublikanoko unitateen arteko gudukak igande gauean hasi ziren Karbala inguruan -musulman xiitentzat hiri santua da-.

Query key:
w61-unitateen

- A-IZE-ARR-182
- A-IZE-ARR-183

- unitateen A-IZE-ARR-182
  - *unitate* «unitate»
    - POS = NOUN
    - SUBCAT = COMMON
  - en «eM»
    - POS = DEC
    - CASE = GEN
    - NUM = PL
    - DEF = DEF
    - SYNTFL = @CN> @<CN

index

- unitateen A-IZE-ARR-183
  - *unitate* «unitate»
    - POS = NOUN
    - SUBCAT = COMMON

Figure 3: Application GUI.

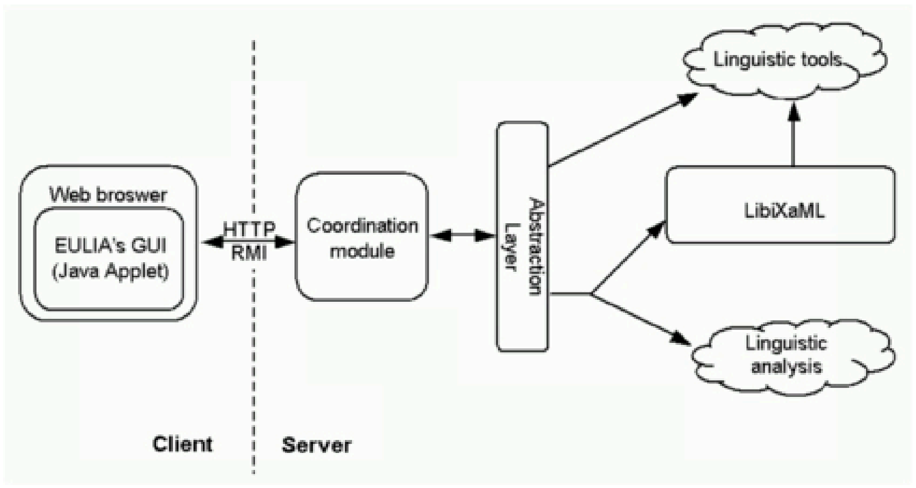


Figure 4: General architecture of the tool.

- Abstraction layer: the main goal of this module is to keep separate the coordination module of integrated tools, the analyses and their location. In order to archive this goal, this layer implements an interface for the coordination module. In this layer the relation between analysis type and tools and the way to recover stored information is defined, and it facilitates the definition of different computing paradigms to determinate the interaction among the linguistic tools. For the moment, a simple serial model has been implemented.

- Set of tools: this set is composed of integrated tools and their outputs. These tools' input and output are coded according to the integration strategy explained before.

In order to integrate a new NLP tool in EULIA system, the input and output of the mentioned tool has to be coded according to integration strategy presented before. Moreover, for a complete integration, it is necessary to define the relation between the new analysis type and tools and the stylesheets used to visualize this analysis. EULIA is a powerful system but it is not complex thanks to the integration strategy. In this strategy, all linguistic information is coded in a similar model, so the treatment of different data is similar. Moreover, EULIA is a generic system and offers many possibilities to be extended to different applications. EULIA is, without a doubt, a useful basis for different areas of linguistic engineering.

4.3. Example

The interface has been designed to be easy-to-use and intuitive. The main window is divided into two parts (see Figure 3): a left MDI panel where the analyzed text is shown to the user, and the right part where linguistic information is shown in an understandable way. The interface provides hypertextual facilities, showing on the right hand-side linguistic information associated to items selected on the left part. The environment is designed as a tool for general users and linguists. The system gives the information the user has asked for about ambiguous units in a lemmatized text. It is important to notice that the item selected can be, in the example, a single word or a multiword expression, since currently the application has been tuned to deal with lemmatization results (actually, the selectability of text chunks depends on the underlying tool the interface is dealing with).

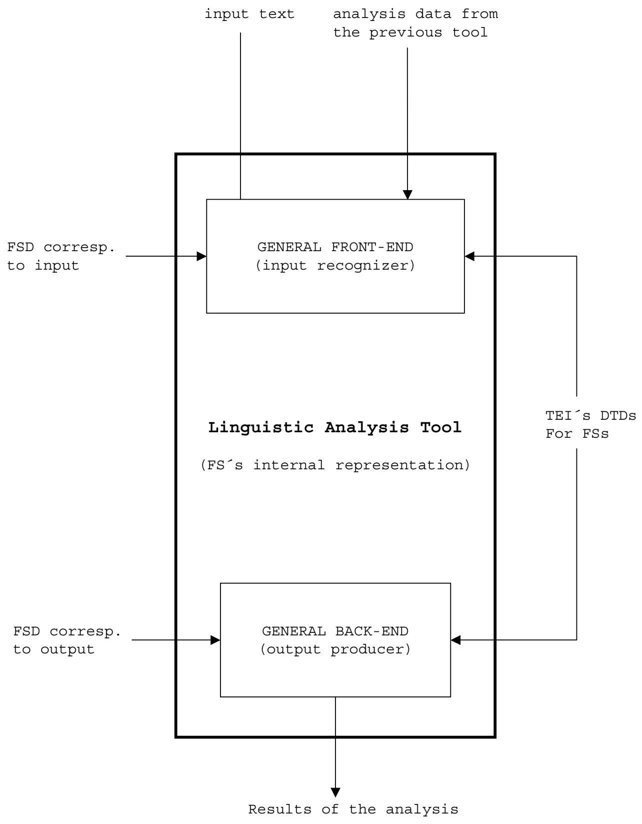

input text
analysis data from the previous tool
FSD corresp. to input
GENERAL FRONT-END (input recognizer)
Linguistic Analysis Tool
(FS´s internal representation)
TEI´s DTDs For FSs
FSD corresp. to output
GENERAL BACK-END (output producer)
Results of the analysis

Figure 5: Schematic view of a linguistic analysis tool with its general front- and back-ends.

5. Conclusion and future work

We have presented a general environment for linguistic processing. The environment is oriented to be used by general users and has been designed to be informative, easy-to-use, and intuitive. It is coupled to a methodology of integration of linguistic tools based on a common annotation framework, general and extensible to similar systems.

For the near future, we are considering the feasibility of building general front- and back-end modules for the analysis tools, which will take as input the specific FSDs for each input/output. A schematic view of the integration of these general modules with a particular tool can be seen in Figure 5. This will facilitate the future integration of new tools into the analysis chain. Indeed, the work done so far confirms the scalability of our approach.

Acknowledgements

This research was partially funded by the Basque Government under the HIZKING21 program (project EJ-ETORTEK2002HIZKING21).

6. References

Aduriz I., Aldezabal J.M., Artola X., Ezeiza N.,Urizar R. 1996. Multiword Lexical Units in EUSLEM: a lemmatiser-tagger for Basque. In *Proc. in Computational Lexicography (Complex'96)*, 1-8. Linguistics Institute, Hungarian Academy of Sciences. Budapest (Hungary).

Aduriz I., Agirre E., Aldezabal I., Alegria I., Ansa O., Arregi X., Arriola J.M., Artola X., Díaz de Ilarraza A., Ezeiza N., Gojenola K., Maritxalar A., Maritxalar M., Oronoz M., Sarasola K., Soroa A., Urizar R., Urkia M. 1998. A Framework for the Automatic Processing of Basque. In *Proc. of the First Int. Conf. on Language Resources and Evaluation*. Granada (Spain).

Aduriz I., Aldezabal I., Ansa O., Artola X., Díaz de Ilarraza A., Insausti J. M. 1998. EDBL: a Multi-Purposed Lexical Support for the Treatment of Basque. In *Proc. of the First Int. Conf. on Language Resources and Evaluation*, vol II, 821-826. Granada (Spain).

Aduriz I., Agirre E., Aldezabal I., Arregi X., Arriola J.M., Artola X., Gojenola K., Maritxalar A., Sarasola K., Urkia M. 2000. A Word-Level Morphosyntactic Grammar For Basque. In *Proc. of the Second Int. Conf. on Language Resources and Evaluation*. Athens (Greece).

Aduriz I., Aldezabal I., Aranzabe M., Arrieta B., Arriola J., Atutxa A., Díaz de Ilarraza A., Gojenola K., Oronoz M., Sarasola K. 2002 Construcción de un corpus etiquetado sintácticamente para el euskera. In *Actas del XVIII Congreso de la SEPLN*. Valladolid (Spain).

Artola X., Díaz de Ilarraza A., Ezeiza N., Gojenola K., Maritxalar M., Soroa A. 2000 A proposal for The Integration of NLP Tools using SGML-Tagged documents. *Second Int. Conf. on Language Resources and Evaluation*. Athens (Greece). May.

Artola X., Díaz de Ilarraza A., Ezeiza N., Gojenola K., Hernández G., Soroa A. 2002. 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 *Third Int. Conf. on Language Resources and Evaluation*. Las Palmas. Spain.

Alegria I., Artola X., Sarasola K., Urkia M. 1996. Automatic morphological analysis of Basque. *Literary & Linguistic Computing*, 11, no. 4, 193-203.

Basili, R., Di Nanni, M., Pazienza, M.T. 1998. "Engineering of IE Systems: An Object-oriented approach". *Information Extraction: Towards scalable, Adaptable Systems*. M.T. Pazienza (Ed.). Springer Verlag.

Bird, S., Day, D., Garofolo, J., Henderson J., Laprun G., Liberman M. 2000. ATLAS: a Flexible and Extensible Architecture for Linguistic Annotation. In *Second Int. Conf. on Language Resources and Evaluation*. 1699-1706. Athens (Greece).

Cunningham H., Gaizauskas R.J. and Wilks Y. 1996. A General Architecture for Language Engineering (GATE) - a new approach to Language Engineering R&D. In *Proceedings of COLING'96*. Copenhagen.

Ezeiza N., Aduriz I., Alegria I., Arriola J.M., Urizar R. 1998. Combining Stochastic and Rule-Based Methods for Disambiguation in Agglutinative Languages. In *Proc. COLING-ACL'98*, 10-14. Montreal (Canada).

Goldfarb, C.F. 1999. *The XML Handbook*. Prentice Hall Iberia. SRL, Madrid.

Ide N., Véronis J. (eds.), 1995. *Text Encoding Initiative. Background and Context.* Kluwer Academic Pub.

Ide N., Romary L., 2003. A Common Framework for Syntactic Annotation . *Proc. of ACL'2001*, pp 298-305. Toulouse (France).

Ide N., Romary L., Clergerie E. de la, 2003. International Standard for a Linguistic Annotation Framework. *Proc. HLT-NAACL 2003 Workshop: Software Engineering and Architecture of Language Technology Systems*, pp 25-30. Edmonton (Canada).

Jacobson R. 1949. The Identification of Phonemic Entities. *Travaux du Cercle Linguistique de Copenhague*, 5, 205-213.

Karlsson F., Voutilainen A., Heikkilä J., Anttila A. 1995. *Constraint Grammar: A Language-independent System for Parsing Unrestricted Text*. Mouton de Gruyter.

Schäffer U. 2003. WHAT: An XSLT-based Infraestructure for the Integration of Natural Language Processing Components. *Proc. HLT-NAACL 2003 Workshop: Software Engineering and Architecture of Language Technology Systems*, pp 9-16. Edmonton (Canada).

Simkins N. K. 1994. An Open Architecture for Language Engineering. In *First CEC Language Engineering Convention*. Paris.

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