Iula2Standoff: a tool for creating standoff documents for the IULACT

Carlos Morell, Jorge Vivaldi, Núria Bel
Institut Universitari de Lingüística Aplicada (IULA)
UPF, Roc Boronat 138, 08018 Barcelona, Spain
E-mail: {carlos.morell, jorge.vivaldi, nuria.bel}@upf.edu

Abstract

Due to the increase in the number and depth of analyses required over the text, like entity recognition, POS tagging, syntactic analysis, etc. the annotation in-line has become unpractical. In Natural Language Processing (NLP) some emphasis has been placed in finding an annotation method to solve this problem. A possibility is the standoff annotation. With this annotation style it is possible to add new levels of annotation without disturbing exiting ones, with minimal knock on effects. This annotation will increase the possibility of adding more linguistic information as well as more possibilities for sharing textual resources. In this paper we present a tool developed in the framework of the European Metanet4u[1] (Enhancing the European Linguistic Infrastructure, GA 270893) for creating a multi-layered XML annotation scheme, based on the GrAF proposal for standoff annotations.

Keywords: text handling, standoff markup, GrAF

1. Introduction

Since the seventies linguistic research has heavily been relied on corpora. They became relevant linguistic resources that permitted to obtain a huge amount of knowledge about language behaviour in real use. A large number of such resources have been compiled by almost all research institutions involved in any area of linguistic research.

At the beginning the resulting material was used without any additional processing; later, corpora have received different levels of processing: different type of segmentation, POS tagging, syntactic analysis, etc. Usually such annotations were added in the same file (in-line annotations), but as the number and depth of analysis has increased such method has became impracticable. The obvious solution is to have a different document for each annotation and consequently a link between the base document and the documents that contains the annotations. This method is known as "standoff annotation". Ideally, annotation information over a text file is physically separated from it, and there is only a reference to it.

The advantages of this annotation style may be summarized as follows:

- It is possible to have annotations with crossing regions (ex. Annotation of the same text from different perspectives)
- New levels of annotation can be added without disturbing existing ones
- It is possible to have several versions of a single annotation type (ex. use different POS taggers on the same text for comparing their results)
- Editing one level of annotation has minimal knock-on effects on others
- Allows distributing the corpus independently of their annotations or selectively annotated.

In spite of the above mentioned advantages its application has been much reduced, mainly due to the complexity of its implementation. Only some language related areas (speech corpora in particular) have taken advantage of this methodology. This kind of annotation is gaining attention because there is an increasing need of implementing multiple annotation levels on the same piece of text.

The purpose of the tool presented in this paper is to perform a basic linguistic processing on free text. These processings are those typical to almost any NLP task: text segmentation, name entity detection and POS tagging. This tool is already available as PANACEA web service (http://registry.elda.org/services/187) and soon will be available as a demo at our Institute's web page (http://www.iula.upf.edu/indexuk.htm).

Following this introduction the paper presents in section 2 a brief overview of the advantages of standoff annotation as well as an outline of already existing implementations. Then, in section 3, we present our two cases of study and in section 4 explain how we designed it. In Section 5 we discuss about the possibility of using byte or chart as basic unit of measurement. After this, in section 6 the evaluation results. Lastly, section 7 introduces some conclusions and future work.

2. State of the art

In corpus compilation the data undergoes a number of processes (text segmentation, lemmatization, POS tagging, etc.) that result in the necessity of some kind of annotation for each component of the corpus. In the case of corpus linguistics, the EAGLES initiative developed a recommendation (CES: Corpus Encoding Standard) about how to incorporate this new information to the text. This recommendation was based on the Text Encoding Initiative, a more general international and interdisciplinary standard for representing all kind of texts. Later, CES has evolved to XCES to keep track of recent encoding standards (XML). Since the very beginning CES foresee two ways to save such extra data: i) inline (a single big file that merges text data with linguistic mark-up data) and ii) standoff (keeping the original text data untouched but adding any extra mark-ups in separate files).

[1] http://www.meta-net.eu/projects/METANET4U

As corpora annotation becomes more and more complex the limitations of the inline model it becomes clear: modifying a level of annotation implies to reprocess completely the text, adding new levels of annotations as well as reusing the annotations from different formats become very complicated. For these reasons, corpus development is changing the way to introduce annotations. Several corpora have been developed using this approach:

- The American National Corpora [2] (ANC) architecture described in Ide et al. (2006) from the very beginning uses the standoff approach to incorporate multiple layers of linguistic data to this resource.
- The GREC 3 (Gene Regulation Event Corpus) corpus (Thompson et.al, 2009). It is a resource in the biomedical field that annotates entities and their relationships.
- Polish National Corpus; see Bański et al., 2009.

The ANC corpus has been produced using annotation tools integrated in GATE and UIMA environments. The ANC2GO is a tool that allows converting ANC standoff files into files with inline annotations in several formats useful with traditional tools. The Polish National Corpus also uses standoff but according to the TEI guidelines.
Another tool available for performing stand-off annotations is MMAX2; see Müller et al. (2006) for details. A main characteristic of this tool is that it allows manual annotation of a corpus using a GUI interface; this fact implies a major difference regarding the tool proposed here. MMAX2 also allows to query the corpus and visualize the results as well as to convert the annotations to several formats.
Due to the difficulties found to integrate already existent tools for Spanish and Catalan we choose to develop our own annotation tool. In our implementation of the standoff approach we imagine two different scenarios:

- To convert to standoff our already existent IULA LSP corpus (Vivaldi, 2009) already tagged following CES standard
- To incorporate standoff capabilities to our text handling tool (Martínez et al., 2010)[4]

In both cases, the text is UTF-8 encoded and the annotations are represented in a generic XML format using GrAF (see section 3.1).

3. Standoff tool

As mentioned above, the main goal of this project is the implementation of an automatic mechanism that given an already existent text handling and POS tagging tool returns a XML file collection according to the GrAF standard. As mentioned above, our environment foresees two different scenarios for obtaining standoff files: i) from our LSP corpus already tagged and ii) from free text. In the following subsections we will briefly describe the behaviour of the tools designed to cover both scenarios mentioned above.

3.1 Project Overview

GrAF is a data model based on graph theory that can be used to represent linguistic annotations and fully described in Ide et al. (2007). It indicates how to describe/represent linguistic annotations in a set of standoff files that later can be analysed using standard graph algorithms and tools.
In practice, it should be considered a *de facto* standard as it is being used in several European projects (Metanet4u and Panacea) as well as in the ANC project.
In our scenarios, the output generated and proposed by this format comprises the following information:

- Structural markup
- Words boundaries
- Words (tokens) with part of speech annotations using the treeTagger (Schmid, 1994)
- Plain text
- Header file

3.2 Inline to standoff conversion

In this case a specific tool was built as the files to be converted are already linguistically processed and have an in-house format using inline tagging. Therefore the only requirement was to produce a number of files reflecting the information already embedded in the input file. Figure 1 shows the diagram for performing this task.

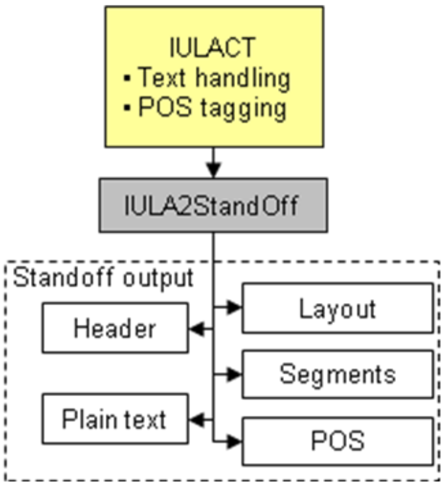


Figure 1 Diagram for converting inline to standoff

The input text is a verticalised text file containing both text handling (sentence delimitation, heading, lists, name entities, foreign words, etc.) and POS tagging information as shown in Listing 1. Therefore, no particular NLP procedure is necessary and a single and *ad hoc* module is able to build the standoff files.
It is important to note that the original plain text used to build the corpus does not exist but it is generated from the verticalised input text.

[2] American National Corpus (www.anc.org)
[3] www.nactem.ac.uk/GREC/
[4] kurwenal.upf.edu/cgi-bin/hectorwww/hectormain.pl

```
##   TAG <div1>
##   TAG <p>
##   TAG <s>
##   TAG <name>
1    TOK It      BOS It\N4666
##   TAG </name>
---  DLI '           =\DELIM
2    PGR s           pr\R6EZZZZ
3    TOK a           a\P
##   TAG <num>
4    TOK 32          num\X
##   TAG </num>
---  DLE -           =\DELIM
5    TOK bit         bit\N5-MS
6    TOK computer         computer\N5-MS
---  DLD .       EOS =\DELS
##   TAG </s>
##   TAG <s>
…
```

Listing 1. Sample of the input text to be converted from inline to standoff

3.3 Update a text handling tool with standoff capabilities

The other scenario foresees to generate a standoff output from plain text. For such purpose, we modify conveniently our already existent text handling tool (see Martínez et al., 2010) for adding standoff capabilities. Figure 2 shows a diagram for performing this task.
In this case, the goal is to improve the already existent text handling tool to include the capability to generate standoff files without any modification to the text handling itself. This behaviour has been achieved by adding one module, modifying other module and generating an auxiliary file (tagged with an '*' in Figure 2).

The segmenter module is responsible for reading the plain text file and to generate the segmentation info; that is, the information about the starting/ending position of each token. Such data are used to generate the segmentation file as required by the GrAF format as well as an additional file required by the output formatting module.

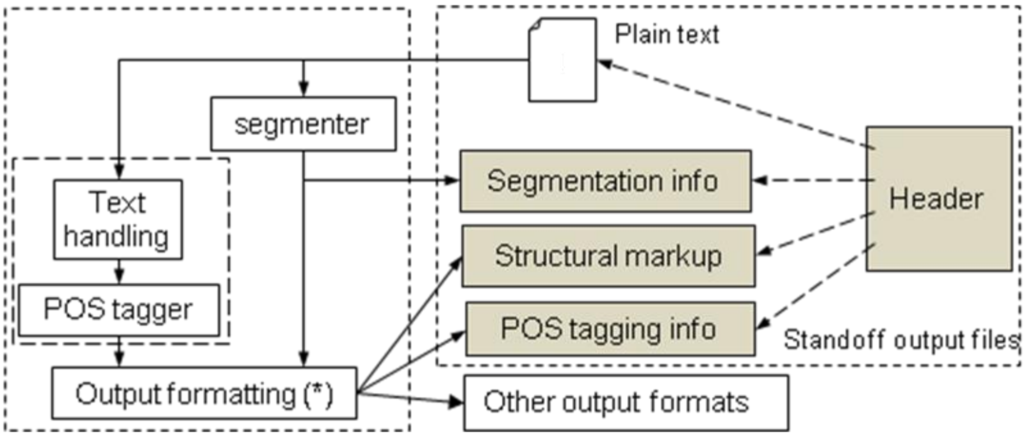


Figure 2 Standoff files creation from plain text

4. Workflows

As mentioned above there are two different scenarios, in the first case we start from a verticalised text and we need to generate all the files while in the second scenario it is necessary to generate all the files with the exception of the input plain text that must remain untouched. In spite of these differences both tools may be split in the following four modules:

- Segmentation
- Tagger
- Structural information
- Header

In the following subsections we will briefly explain the behaviour of each module in each scenario.

4.1 Segmentation

This module is responsible for dividing the full text in segments, which means to establish the start/end position of each region and token. Conventionally we state that a region is any sequence of characters surrounded by one or more blank spaces and a token is a single linguistic unit. Therefore a single word (like *mesa* –table-) normally is a region and also a token but there some circumstances where a single word may be two tokens (Spanish contractions like *del* = *de* + *el* –of the–) or two or more words form a single token (like many name entities like *Buenos Aires*).
According to the input text there are two different situations:

- Plain text

We must calculate such positions from the input text file. Each word or punctuation mark will become a region. Later, the text handling will decide if such region will become or not a token. No other special care should be taken.

- Already verticalised file

In this case the text has been already tokenised; therefore, we must proceed in a different way. Every token will become always a region but we must also adopt some conventions. In particular we must consider that every token is preceded by a single white space. Also, special care must be taken with already existent structural markup (paragraph, sentence, heads, etc.), punctuation marks

must be taken into account and special care should be taken with some conventions marks (EOS, SENT, BOS, etc.) for generating a proper plain text.
The above mentioned rule is true in most cases, but not when a word is at the beginning of the sentence or it is followed/preceded by one or more punctuation marks.
The structural markups <head> and <p> (and their respective </head> and </p>) indicate where header and paragraph starts and ends. When this marks appear it is necessary to introduce new lines and the following word will be a beginning of sentence, for that reason it is not necessary the white space in front it. Something similar happens with <s> and </s> (start/end of sentence) marks.
The punctuation marks have a special particularity; the verticalised test have a signal that indicates how their positioned between their predecessor and their subsequent token. There are different marks, each one requiring a different behaviour:

- DLE: white space in the left side
- DLD: white space in the right side
- DLS: isolated punctuation mark (white space in both sides)
- DLI: inner punctuation mark (no white space in both sides, like the apostrophe "" in the Catalan expression *l'arxiu* –the archive–)

The EOS or SENT elements, indicate the ends of sentences. This is an important point because is necessary to correctly update the tokens offsets to follow the segmentation marking.
Other elements that indicate a special behaviour is the PGR elements. This tag indicates that such token must be hooked to its predecessor.

4.2 Tagging information

This module is responsible of connecting POS information provided by the tagger with the segments and tokens creating the corresponding file.
Again both scenarios create situations a little bit different:

- Plain text

Token information is obtained from the already existent text handling software. At the same time, the segmenter has created the segment information. Therefore, the output formatting module needs to put in correspondence both information and create the appropriate standoff files (see Figure 2) taking care to keep track of the offset data.
In Figure 3 we show the relationship established between the plain text file and the produced standoff files. The word "this" for example, the segmented modules produces a region (identified as "seg-r0") and the POS tagger identifies it as a pronoun (giving both lemma and tag "RDS3N-"). Both are combined in the POS file that put in relation the word form in the plain text file with the corresponding linguistic data.

- Already verticalised file

Token information has already been calculated but we do not have region information. It must be created in accordance to the token information. In this case is important to notice that a relation among tokens and regions is not always 1:1. It may happen the case where a single token corresponds to several regions. See for example the following date: "23 de mayo del 2012". It has been already identified as a single token by the existent text handling software ant the POS information has the following form:

```
 ##  TAG  <date>
 123 TOK     23 de mayo del 2012 =\W
 ##  TAG  </date>
```

From this information it is necessary to generate 5 regions and create a single node with the lemma and POS data.
Conversely, it may also happen that a single word, generate two tokens and two regions. See for example the Spanish contraction "del". The POS tagger provides the following information:

```
 123 TOK  de       de\P
 124 PGR  l        el\AMS
```

From these lines, we may deduce that it is necessary create two nodes, each one with the corresponding region, lemma and POS data.

4.3 Structural information

The purpose of this module is to find the regions that correspond to the main structural information (sections, heads, paragraphs, lists, etc.) as well as some intratextual markers (named entities: proper nouns, date, numbers, etc.). From the operating point of view, this module is identical in both scenarios although their implementation is a little bit different.
All textual and intratextual structures are delimited by their own XML tags, like <s>...</s>, <date>...</date> and similar. These structures are composed by N segments, inside the delimiting tags. In the segmentation phase we create a relation between every segment and its relative position in the text. It is natural that we consider that the first position of the structure will be the same that the first position of the first segment and same with the last position. Then just keeping a careful counter of segment we can create the structure layout of the text.

4.4 Header

This module is responsible for the creation of a header file. It contains the links to all the files associated to the text. It also contains some bibliographic information about the author, the date of publication, the edition of the text and other relevant information referring to the text.
In this case a differentiation between the two scenarios is not necessary although the available information may be very different. The information from documents from the IULA LSP corpus is very exhaustive while those associated to plain text may be very succinct. In any case we use the same module to do it.
We will maintain an exhaustive control on the modifications of the files including a version control in order to know which was the last modification and who has done it.

5. Offset information

Essential information associated with the standoff is the offset data, which is the displacement of any relevant piece of information regarding the beginning of the file. GrAF specification states that it should be indicated in characters. We implemented in this way and it works fine in English text but shows some troubles in Spanish/Catalan text. In English texts it works correctly because they use ISO encoding and therefore there is

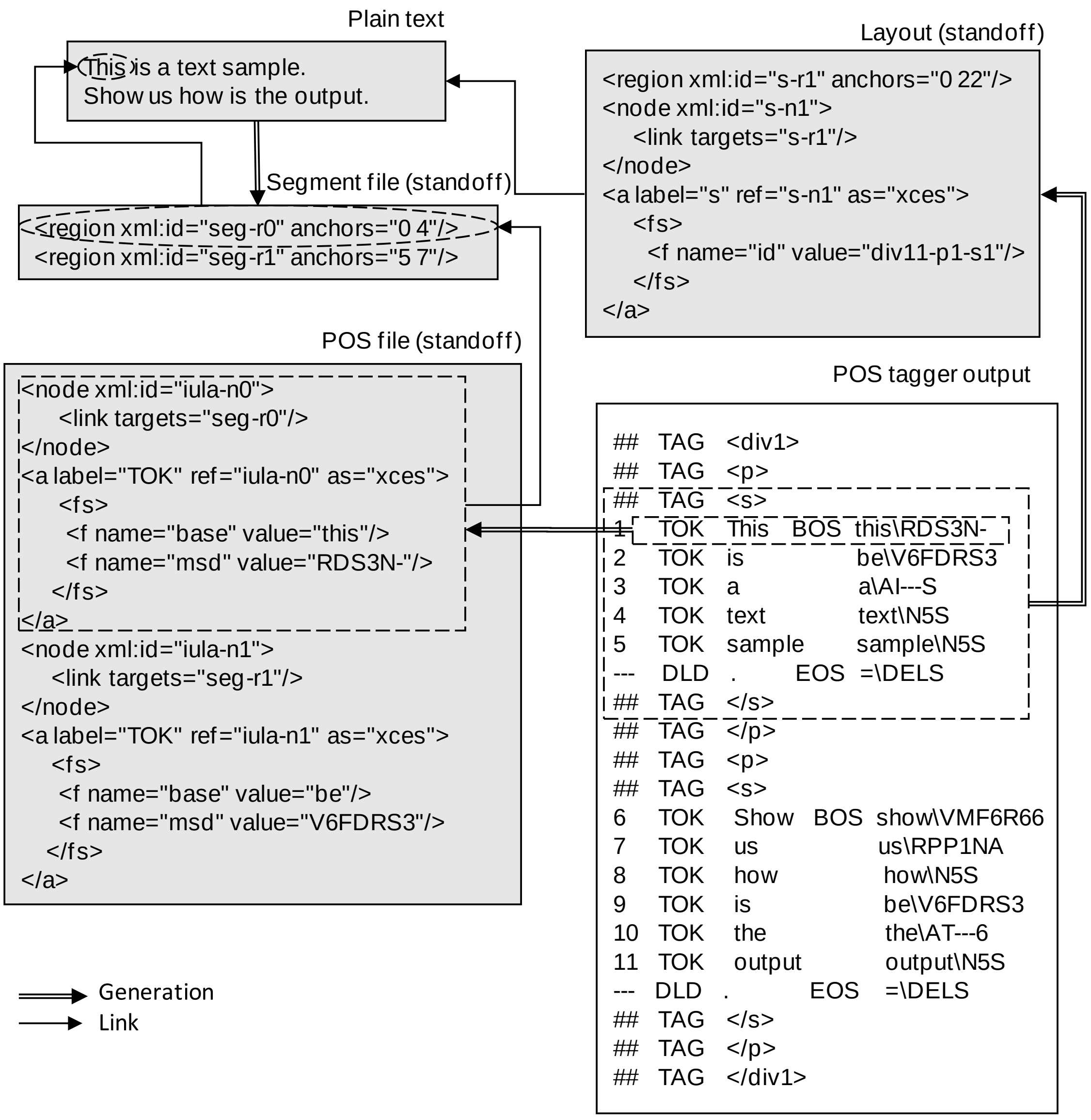
always a correspondence 1:1 among characters and bytes. But languages like Spanish or Catalan may require two bytes for some single characters (Ç, ü, ó, ñ, etc.).
Therefore an efficiency problem arises because most of programming languages may access to a given file position more efficiently when such position is indicated by bytes instead of characters.
To verify such condition, we have performed some tests reading text files (of several lengths) where the position of every word has been indicated by characters or by bytes.
For reading we use some Perl script (for the offset in bytes or chars) and Java programs (chars only). The test consists of the reading all the tokens of the text and the resulting figures are the following:

File Size	Bytes(Perl)	Char(Perl)	Char(Java)
32 kB	0.45s	135s	1.9s
64 kB	0.92s	1185s	5.0s
127 kB	1.18s	2256s	15.7s

As foreseed, the above table shows that indicating the offsets in bytes, the reading is much more efficient and such benefit increases with the length of the file. The access by chars using Perl may be optimized for sequential access but this may not be the general case. It is assumed that byte reading using Java should be at least as efficient as Perl.

6. Evaluation

In any case the final result is the creation of a set of files with standoff annotation. Figure 3 shows both the logic links among every file and how the information is obtained from the POS tagger output and/or the input text. The segment file is generated indicating for each segment its position and assigning a unique identifier. The morphosyntactic information is indicated in the POS file which obtains this information from the POS tagger output.



```
This is a text sample.
Show us how is the output.
```

```
<region xml:id="s-r1" anchors="0 22"/>
<node xml:id="s-n1">
    <link targets="s-r1"/>
</node>
<a label="s" ref="s-n1" as="xces">
    <fs>
      <f name="id" value="div11-p1-s1"/>
    </fs>
</a>
```

```
<region xml:id="seg-r0" anchors="0 4"/>
<region xml:id="seg-r1" anchors="5 7"/>
```

```
<node xml:id="iula-n0">
     <link targets="seg-r0"/>
</node>
<a label="TOK" ref="iula-n0" as="xces">
     <fs>
      <f name="base" value="this"/>
      <f name="msd" value="RDS3N-"/>
    </fs>
</a>
<node xml:id="iula-n1">
    <link targets="seg-r1"/>
</node>
<a label="TOK" ref="iula-n1" as="xces">
    <fs>
     <f name="base" value="be"/>
     <f name="msd" value="V6FDRS3"/>
   </fs>
</a>
```

```
##  TAG  <div1>
##  TAG  <p>
##  TAG  <s>
1   TOK  This   BOS  this\RDS3N-
2   TOK  is          be\V6FDRS3
3   TOK  a           a\AI---S
4   TOK  text        text\N5S
5   TOK  sample      sample\N5S
--- DLD  .      EOS  =\DELS
##  TAG  </s>
##  TAG  </p>
##  TAG  <p>
##  TAG  <s>
6   TOK  Show   BOS  show\VMF6R66
7   TOK  us          us\RPP1NA
8   TOK  how         how\N5S
9   TOK  is          be\V6FDRS3
10  TOK  the         the\AT---6
11  TOK  output      output\N5S
--- DLD  .      EOS  =\DELS
##  TAG  </s>
##  TAG  </p>
##  TAG  </div1>
```

Figure 3. Linking among files

This structure become rather complicated as the input file becomes larger and larger; therefore, manual checking becomes impossible by humans. For such purpose we developed some checking scripts that check:

- Coherence among segment/POS file and input text,
- Coherence among stand-off files and inline files (only for inline to standoff conversion).

The second test is the only that is fully exhaustive because we have the inline file that may be taken as a reference. It allowed, as a side effect, the production of a general purpose Perl library to access these files.

7. Conclusions and Further work

This paper presented a tool to generate multi-layered XML annotation scheme from an already existent corpus as well as generate such files from free text.

Although the current definition of standoff files work properly, during the evaluation of the tool we find some difficulties to work with the layout file. Both the textual structure and word regions are defined through anchors related to plain text. It seems that should simpler if textual structures are referred to the word regions already defined in the segment file.

All the NLP tools developed and in use in our Institute use the inline annotation. We plan to improve such tools in order to take advantage from this output. For example, our term extractor may generate an additional layer to indicate which are the terms occurring in a given text. Another application that will apply this annotation technique will be the Spanish syntactic parser currently under development.

8. Acknowledgements

This work has been funded by the EU project 270893 (Metanet4U: Enhancing the European Linguistic Infrastructure).

9. References

Bański P. and A. Przepiórkowski (2009). *Stand-off TEI Annotation: the Case of the National Corpus of Polish.* In: the proceedings of the LAW-III workshop at ACL-IJCNLP 2009, Singapore.

Ide, N., Suderman, K. (2006). Integrating Linguistic Resources: The American National Corpus Model Proceedings of the Fifth Language Resources and Evaluation Conference (LREC), Genoa, Italy.

Ide, N., Suderman, K. (2007). GrAF: a graph-based format for linguistic annotations. Proceedings of the Linguistic Annotation Workshop, ACL, pages 1–8, Prague

Martínez, Héctor; Vivaldi, Jorge; Villegas, Marta (2010). "Text handling as a Web Service for the IULA processing pipeline" in Calzolari, Nicoletta et al. Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10). Paris, France. Pp. 22-29.

Müller, C.; Strube m. (2006): Multi-Level Annotation of Linguistic Data with MMAX2. In: Sabine Braun, Kurt Kohn, Joybrato Mukherjee (Eds.): Corpus Technology and Language Pedagogy. New Resources, New Tools, New Methods. Frankfurt: Peter Lang, pp. 197-214.

Schmid, Helmut (1994): Probabilistic Part-of-Speech Tagging Using Decision Trees. *Proceedings of International Conference on New Methods in Language Processing*, Manchester, UK.

Thompson, P., Iqbal, S. A., McNaught, J. and Ananiadou, S. (2009). Construction of an annotated corpus to support biomedical information extraction. BMC Bioinformatics 10:349

Vivaldi, Jorge (2009). "Corpus and exploitation tool: IULACT and bwanaNet" en Cantos Gómez, Pascual; Sánchez Pérez, Aquilino (ed.) *Proceeding of the first International Congress of Corpus Linguistics (CICL-09)* Murcia. Spain. Pp. 224-239.