recibido 18-05-2007; aceptado 22-06-2007

Building Corpora for the Development of a Dependency Parser for Spanish Using Maltparser[*]

Jesús Herrera
Departamento de Lenguajes y Sistemas Informáticos
Universidad Nacional de Educación a Distancia
C/ Juan del Rosal, 16, E-28040 Madrid
jesus.herrera@lsi.uned.es

Pablo Gervás, Pedro J. Moriano, Alfonso Muñoz, Luis Romero
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid
C/ Profesor José García Santesmases, s/n, E-28040 Madrid
pgervas@sip.ucm.es, {pedrojmoriano, alfonsomm, luis.romero.tejera}@gmail.com

Resumen: En el presente artículo se detalla el proceso de creación de corpora para el entrenamiento y pruebas de un generador de analizadores de dependencias (Maltparser). Se parte del corpus Cast3LB, que contiene análisis de constituyentes de textos en español. Estos análisis de constituyentes se transforman automáticamente en análisis de dependencias. Además se describe cómo se obtiene, experimentalmente y de manera semiautomática, un conjunto de etiquetas de funcionalidad sintáctica para etiquetar adecuadamente el corpus de entrenamiento. El proceso seguido ha permitido obtener un analizador de dependencias para el español con una precisión del 91 % en la determinación de dependencias.
Palabras clave: Análisis de dependencias, corpus de entrenamiento, etiqueta de funcionalidad sintáctica, Maltparser, JBeaver

Abstract: The present paper details the process followed for creating training and test corpora for a dependency parser generator (Maltparser). The starting point is the Cast3LB corpus, which contains constituency analyses of Spanish texts. These constituency analyses are automatically transformed into dependency analyses. In addition, the empirically and semiautomatically obtention of a set of syntactic function labels for the training corpus is described. As a result of the process followed, it has been obtained a dependency parser for Spanish showing a 91 % precision when determining dependencies.
Keywords: Dependency parsing, training corpus, syntactic function label, Maltparser, JBeaver

1. *Introduction*

The development of JBeaver, a dependency parser for Spanish (Herrera et al., 2007), is based on the use of Maltparser (Nivre et al., 2006), which is a machine learning tool for generating dependency parsers for, virtually, every language. Such development carries inherently associated the labour of generating corpora for its training and its subsequent evaluation.

The amount of work needed for developing from scratch a corpus annotated with dependency analyses, and with a suitable size for training Maltparser, exceeded the possibilities of the JBeaver project. Therefore, it was necessary to find an alternative way for the generation of such corpus. A possible approach was to reuse available resources in order to build from them a corpus annotated with dependency analyses in a semiautomatic way. For this, the Cast3LB (Navarro et al., 2003) treebank was used. It is conformed by 72 Mb of Spanish annotated texts, approximately and itcontains the constituency analysis for every sentence in it. Leaving

[*] Partially supported by the Spanish Ministry of Education and Science (TIN2006-14433-C02-01 project).

aside certain subtleties (Gelbukh and Torres, 2006), constituency analysis and dependency analyses can be converted one into the other in a systematic way. After studying the format and labels used for Cast3LB (Navarro et al., 2003) (Civit, 2002), a system capable of transforming the constituency analyses contained in Cast3LB into dependency analyses was developed by modifying an algorithm proposed by Gelbukh et al. (Gelbukh and Torres, 2006) (Gelbukh et al., 2005). The existence of Cast3LB and the possibility of transforming the analyses contained in it into dependency analyses were important reasons to use Maltparser in the JBeaver project.

On the other hand, having decided that the JBeaver parser would be made generally available to the public, lead us to consider additional requirements. For instance, we decided to make as easy as possible the use of JBeaver by tools already adapted to the use of Minipar (Lin, 1998). This is due to the fact that Minipar has become a *de facto* standard in the last years after being used by a large number of applications. Thus, the notation used for JBeaver is, as far as possible, the same as the one used for Minipar.

2. The source corpus

A dependency analysis corpus is needed for training Maltparser. The construction of such a corpus by hand implied a work load well beyond the constraints of the JBeaver project. Thus, it was decided to take advantage of existing resources. Taking into account that, except for some specific cases (such as non-projective constructions), the dependency analysis of a text can be automatically derived from its constituency analysis (Gelbukh and Torres, 2006), and that Cast3LB –which contains constituency analyses of Spanish texts– was available, it became the best option as source corpus for the project. Then, the training corpus was obtained in a semiautomatic way from Cast3LB.

Cast3LB contains 100,000 words in, approximately, 3,700 sentences of texts in Spanish. 75,000 words of Cast3LB come from the ClicTALP corpus, which is a set of text from several domains: literary, journalistic, scientific, etcetera, and the other 25,000 words come from the EFE news agency's corpus from year 2000 (Navarro et al., 2003). In figure 1 an excerpt from Cast3LB is shown as an example.

3. Building a training corpus

Malparser requires for its training a corpus in which, for every word of the analyzed text, the following data must be incorporated: a unique identifier, its part of speech label, the identifier of the head of that word and a label indicating the syntactic function given in the dependency relationship. Maltparser admits both a XML format and a tab format at its input. In figure 2 two mutually equivalent examples are shown (the first one in XML format and the second one in tab format).

The numeric identifier 0 and the syntactic function label ROOT are used by convention to designate the dependency tree's root[1].

All the information needed for the creation of the training corpus was contained in the Cast3LB corpus, but it was necessary to extract it and to modify it to suit the conventions followed by Maltparser. For this, the two following actions were accomplished: the obtention of dependency relationships, and the obtention of syntactic function labels.

3.1. Obtaining dependency relationships

In order to extract the dependency relationships between words contained in the Cast3LB corpus, an automatic process was developed. It was designed from an algorithm proposed by Gelbukh et al. (Gelbukh and Torres, 2006) (Gelbukh et al., 2005), modified as needed.

3.2. Obtaining syntactic functions labels

The great popularity reached in the last years by Minipar lead to the decision of using, in the JBeaver project, a set of syntactic function labels that followed, as far as possible, the nomenclature given by Minipar. In this way, it would be easier to adapt systems currently using Minipar to the use of JBeaver. Since the Cast3LB corpus contains specific syntactic function labels, they must be translated into the ones used by Minipar in order to train Maltparser with the appropriate set of labels. For this, the first action to be accomplished was to obtain the set of syntactic function labels from Minipar. Since

[1]`http://w3.msi.vxu.se/~nivre/research/MaltXML.html`

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE FILE SYSTEM "3lb.dtd">
<FILE id="agset" language="es" wn="1.5" ewn="dic2002"
 parsing_state="process" semantic_state="process"
 last_modified="13-01-2006" project="3LB" about="3LB project annotation file">
<LOG auto_file="a1-0-auto3.log" anno_file="a1-0-anno4.log"
 nosense_file="a1-0-nosense4.log" />
<SENTENCE id="agset_1">
<Anchor id="agset_1_ac1" offset="0"/>
<Anchor id="agset_1_ac2" offset="15"/>
<Anchor id="agset_1_ac3" offset="21"/>
<Anchor id="agset_1_ac4" offset="23"/>
<Anchor id="agset_1_ac5" offset="26"/>
<Anchor id="agset_1_ac6" offset="34"/>
<Anchor id="agset_1_ac7" offset="40"/>
<Anchor id="agset_1_ac8" offset="42"/>
<Anchor id="agset_1_ac9" offset="52"/>
<Anchor id="agset_1_ac10" offset="54"/>
<Annotation id="agset_1_an3" start="agset_1_ac1" end="agset_1_ac2"
 type="syn">
<Feature name="roles">SUJ</Feature>
<Feature name="label">sn</Feature>
<Feature name="parent">agset_1_an2</Feature>
</Annotation>
<Annotation id="agset_1_an4" start="agset_1_ac1" end="agset_1_ac2"
 type="syn">
<Feature name="label">grup.nom.ms</Feature>
<Feature name="parent">agset_1_an3</Feature>
</Annotation>
<Annotation id="agset_1_an5" start="agset_1_ac1" end="agset_1_ac2"
 type="wrd">
<Feature name="label">Medardo_Fraile</Feature>
<Feature name="sense">C2S</Feature>
<Feature name="parent">agset_1_an6</Feature>
</Annotation>
<Annotation id="agset_1_an6" start="agset_1_ac1" end="agset_1_ac2"
 type="pos">
<Feature name="lema">Medardo_Fraile</Feature>
<Feature name="label">np00000</Feature>
<Feature name="parent">agset_1_an4</Feature>
</Annotation>
<Annotation id="agset_1_an1" start="agset_1_ac1" end="agset_1_ac10"
 type="dummy_root">
<Feature name="label"/>
<Feature name="parent"/>
</Annotation>
```

Figura 1: Excerpt from Cast3LB

an exhaustive list of these labels is not publicly available, it was necessary to try to obtain the best possible approach, from a large number of analyses made with Minipar. Following this goal, an empirical work was accomplished, based on the idea that with a great amount of analyses made with Minipar the set of different labels found would be very close to the real set of labels. The process employed was the following:

```
<sentence id="2" user="malt" date="">
  <word id="1" form="Genom" postag="pp" head="3" deprel="ADV"/>
  <word id="2" form="skattereformen" postag="nn.utr.sin.def.nom" head="1"
   deprel="PR"/>
  <word id="3" form="infors" postag="vb.prs.sfo" head="0" deprel="ROOT"/>
  <word id="4" form="individuell" postag="jj.pos.utr.sin.ind.nom" head="5"
   deprel="ATT"/>
  <word id="5" form="beskattning" postag="nn.utr.sin.ind.nom" head="3"
   deprel="SUB"/>
  <word id="6" form="(" postag="pad" head="5" deprel="IP"/>
  <word id="7" form="sarbeskattning" postag="nn.utr.sin.ind.nom" head="5"
   deprel="APP"/>
  <word id="8" form=")" postag="pad" head="5" deprel="IP"/>
  <word id="9" form="av" postag="pp" head="5" deprel="ATT"/>
  <word id="10" form="arbetsinkomster" postag="nn.utr.plu.ind.nom" head="9"
   deprel="PR"/>
  <word id="11" form="." postag="mad" head="3" deprel="IP"/>
</sentence>
```

```
Genom           pp                     3        ADV
skattereformen  nn.utr.sin.def.nom     1        PR
infors          vb.prs.sfo             0        ROOT
individuell     jj.pos.utr.sin.ind.nom 5        ATT
beskattning     nn.utr.sin.ind.nom     3        SUB
(               pad                    5        IP
sarbeskattning  nn.utr.sin.ind.nom     5        APP
)               pad                    5        IP
av              pp                     5        ATT
arbetsinkomster nn.utr.plu.ind.nom     9        PR
.               mad                    3        IP
```

Figura 2: Mutually equivalent training files for Maltparser (XML and tab)

1. A set of English texts obtained from the web was parsed with Minipar. It consisted of about 1 Mb of texts from several domains extracted from the Project Gutemberg[2] covering the following domains: sport (197.1 Kb containing 1,854 phrases), economy (207.1 Kb containing 1,173 phrases), education (160.5 Kb containing 869 phrases), history (162.2 Kb containing 1,210 phrases), justice (98.2 Kb containing 453 phrases) and health (265.2 Kb containing 2,409 phrases).

2. The output files given by Minipar were treated in order to extract the set of all different syntactic function labels.

3. A set of analyses, in which all the labels found were present, was selected and the following algorithm was applied to it:

[2]http://www.gutenberg.org/

```
for each syntactic function label identified do
    if this function may occur in Spanish then
        Set one or more rules for suitably transforming the syntactic function label from Cast3LB into the identified label;
    else
        Discard the identified label;
    end if
end for
```

The rules mentioned above were implemented in the program that transforms constituency analyses into dependency analyses. A special label was used to identify not yet discovered syntactic functions that might be found in the future.

After the establishment of the set of syntactic rules, a significant set of constituency analyses was transformed into dependency analyses. Having obtained the dependency treebank, all the analyses containing one or more special labels for not yet discovered syntactic functions was manually analyzed. Then, every case was studied in order to determine if a new syntactic function label was incorporated to the set or the considered syntactic function could be assimilated to one of the known labels. In figure 3 the complete list of syntactic function labels is shown, i.e., those from Minipar and those that were defined *ad–hoc*.

Identified Minipar's syntactic function labels:

sc	neg	pcomp–n
pnmod	nn	gen
poss	lex–dep	appo
whn	mod	subj
aux	amod	guest
num	vrel	else
punc	det	neg
amount–value		

New *ad–hoc* syntactic function labels:

ROOT	adj	fecha
descr	c-descr	compdet

Figura 3: Syntactic function labels used in the training corpus

The set of syntactic function labels finally obtained was not necessarily complete, but it was reasonably valid for its purpose. Thus, it was used by the algorithm that transformed constituency analyses into dependency analyses for labelling the syntactic functions according to Minipar's nomenclature.

3.3. Part of speech tagging

One of JBeaver's features is that is capable to parse texts with no need of a previous annotation. Since the model learned by MaltParser requires, for the parsing step, that every word is labeled with its part of speech, the tagging subtask is implemented in JBeaver by the part of speech tagger Treetagger (Schmid et al., 1994). The use of Treetagger was motivated by the fact that its set of part of speech labels was the one used for MaltParser's training.

3.4. The definitive corpus

Following the process described in this section, 280 XML files (72.9 Mb) containing constituency analyses from the Cast3LB corpus, consisting of 97,002 words, were transformed into dependency analyses apt for their processing by MaltParser (a tab training file of 1.6 Mb), being labeled according to the requirements of the JBeaver project.

4. The test corpus and results obtained

For the evaluation of the trained model a fraction of dependencies correctly found and labeled was computed. The gold standard was a fraction of the corpus described in section 3. This corpus was divided in three equal parts; two of them were used as the training corpus and the other one was used both as test corpus and as gold standard. For using it as test corpus, the annotations concerning dependency relationships and syntactic function were eliminated, i.e., it was conformed only by the words and their part of speech tags, which is the format required by MaltParser for using it as parser. Thus, the output given by the trained model was compared with the gold standard, and 91 % of the dependencies found by the trained model were according to the gold standard (Herrera et al., 2007). This result is comparable to the one obtained by Nivre et al. when training MaltParser for Spanish (Nivre et al., 2006).

5. Conclusions and future work

The process of building corpora for training and testing a specific tool for generating dependency parser (Maltparser) has been shown. This process has proper features because of the requirements of the project in which it has been developed (JBeaver). It was mandatory to use existing resources, and a constituency analyses corpus has been satisfactorily transformed into a equivalent dependency analyses corpus. For this purpose, an algorithm previously proposed by Gelbukh et al. was modified and applied. In addition and in order to fulfill the necessities of the project, the set of syntactic function labels of Minipar was empirically determined.

The future work includes the search for more syntactic function labels, from Minipar and new ones not considered yet. Also, some research could be done in order to improve the algorithm that transforms constituency analyses into dependency analyses. By means of these future improvements, it should be possible to learn better models for dependency parsing in Spanish.

In addition, similar development efforts to the one described here could be carried out for other languages.

Bibliografía

M. Civit. 2002. *Etiquetación de los Cuantificadores: Varias Propuestas.* TALP Research Center–Universidad Politécnica de Cataluña. Technical Report.

A. Gelbukh and S. Torres. 2006. *Tratamiento de Ciertos Pronombres y Conjunciones en la Transformación de un Corpus de Constituyentes a un Corpus de Dependencias.* Avances en la Ciencia de la Computación. VII Encuentro Internacional de Computación ENC'06.

A. Gelbukh, S. Torres and H. Calvo. 2005. *Transforming a Constituency Treebank into a Dependency Treebank.* Procesamiento del Lenguaje Natural, N^o 35, September 2005. Sociedad Española para el Procesamiento de Lenguaje Natural (SEPLN).

J. Herrera, P. Gervás, P.J. Moriano, A. Muñoz, L. Romero. 2007. *JBeaver: Un Analizador de Dependencias para el Español Basado en Aprendizaje.* Under evaluation process for CAEPIA 2007.

D. Lin. 1998. *Dependency–based Evaluation of MINIPAR.* Proceedings of the Workshop on the Evaluation of Parsing Systems, Granada, Spain.

B. Navarro, M. Civit, M.A. Martí, R. Marcos, B. Fernández. 2003. *Syntactic, Semantic and Pragmatic Annotation in Cast3LB.* Proceedings of the Shallow Processing on Large Corpora (SproLaC), a Workshop on Corpus Linguistics, Lancaster, UK.

J. Nivre, J. Hall, J. Nilsson, G. Eryiğit and S. Marinov. 2006. *Labeled Pseudo–Projective Dependency Parsing with Support Vector Machines.* Proceedings of the CoNLL-X Shared Task on Multilingual Dependency Parsing, New York, USA.

H. Schmid. 1994. *Probabilistic Part-of-Speech Tagging Using Decission Trees.* Proceedings of the International Conference on New Methods in Language Processing, pages 44–49, Manchester, UK.