Procesamiento del Lenguaje Natural, núm. 35 (2005), pp. 145-152 recibido 30-04-2005; aceptado 01-06-2005

Transforming a Constituency Treebank into a Dependency Treebank

Alexander Gelbukh, Sulema Torres, Hiram Calvo
Center for Computing Research, National Polytechnic Institute,
Av. Juan Dios Batiz s/n col. Zacatenco, 07738, DF, Mexico
gelbukh@gelbukh.com, likufanele@likufanele.com, sulema7@hotmail.com

Resumen: Presentamos una técnica heurística para convertir un corpus anotado sintácticamente dentro del formalismo de constituyentes, a un corpus anotado dentro del formalismo de dependencias. Particularmente comentamos sobre nuestra experiencia en convertir el corpus Cast3LB del español. El método consiste en extracción de una gramática libre de contexto del corpus etiquetado, identificación automática del elemento rector en cada regla, y usando esta información para la construcción del árbol de dependencias. Nuestras heurísticas identifican el elemento rector de las reglas con precisión de 99% y cobertura de 80%, con lo que el algoritmo identifica correctamente 92% de las relaciones de dependencias entre las palabras.
Palabras clave: Corpus anotados sintácticamente, constituyentes, dependencias, formalismos gramaticales

Abstract: We present a heuristic technique for converting a constituency treebank into a dependency treebank. In particular, we comment on our experience in converting the Spanish treebank Cast3LB. We extract a context-free grammar from the treebank, automatically identify the head in each rule, and use this information for constructing the dependency tree. Our heuristics have 99% precision and 80% recall in identifying the head in the rules, which gives 92% accuracy in identifying dependencies between words.
Keywords: Treebanks, constituency, dependency, grammar formalisms

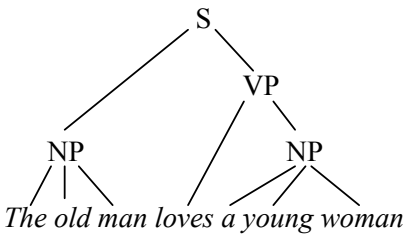
1 Introduction

There are two main formalisms for representing the syntactic structure of a sentence: constituency (or phrase structure) and dependency. Both types of grammars use trees to represent the structure of a phrase, though the meaning of the nodes and links in the tree is different.

In the phrase structure grammar, the nodes of the tree are text spans and the links stand for inclusion relation, e.g.:

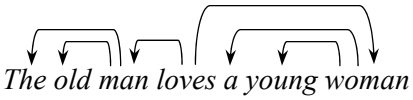
$[[\textit{The old man}]_{NP}\ [\textit{loves}\ [\textit{a young woman}]_{NP}]_{VP}]_{S}$

or, in a graphical form:

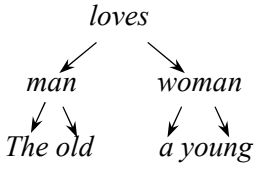


The nodes are labeled: NP stands for noun phrase, VP for verb phrase, S for the whole sentence.

In dependency tree, the nodes of the tree are single words, so that a dependency is established between a pair of words: one of the words is the main or governing, and the other is a subordinate (or dependant) of the first one.



or, in a graphical form:



Each but one word in the sentence has a governing word. A dependency relation be-

ISSN: 1135-5948

tween governor *G* and dependent *D* means, roughly, that a word combination *G D* (or *D G*) is meaningful and inherits the syntactic and semantic properties of *G* (and not *D*): *G D* is a *G* (and is not a *D*); *D* is said to modify *G*. In our example, the combination *old man* is a kind of *man* (and not a kind of *old*); *man loves woman* is a kind of (situation of) *love* (and not, say, a kind of *woman*). Unlike phrase structure tree, in the dependency tree the arcs are (or can be) labeled: *old* and *young* are attribute modifiers of *man* and *woman*, respectively; *man* is the subject of *loves*.

Dependency representation greatly simplifies certain tasks as compared with constituency approach. For example:

– In lexicography, gathering statistics on syntactic combinability of individual words (*read a book* and *hammer a nail* vs. **read a nail* and **hammer a book*) is trivial over dependency representation: one just counts the frequencies of arcs connecting the instances of two given words in the corpus. One of numerous applications of such statistics (Bolshakov, 2004; Bolshakov and Gelbukh, 2001, 2003) is syntactic disambiguation: the tree with frequently combined word pairs is preferred (Yuret, 1998; Gelbukh, 1999). With phrase structure approach this is difficult to impossible.
– In information retrieval and text mining, matching phrase or complex query with the sentences in the corpus is, again, nearly trivial over dependency tree: a query *shirt with long sleeves and red strips* will easily compare with a description *A shirt of high-quality silk with red wide vertical strips and long blue sleeves* in an e-commerce database, but now with *A red shirt with long blue strips on the sleeves*.
– In semantic analysis, transforming the dependency tree in nearly any semantic representation—such as conceptual graphs (Sowa, 1984) or semantic network (Mel'čuk, 1998)—is much more straightforward. In fact, HPSG builds a kind of dependency tree to construct its minimal recursion semantic representation (Sag *et al*., 2003).

However, most (tough not all) of existing tools and resources are oriented to phrase-structure representation, for which is (arguably) simpler to build a parser.

In spite of apparent differences, both representations share the bulk of information on the syntactic structure—to such degree that they can be combined (Sag *et al*., 2003) one can be automatically derived from the other, given some information is added that is present in the second representation but absent in the first one. Basically, phrase structure bears more information on the word order within a structural unit, while a dependency structure bears more information on inheritance of syntactic properties within such unit.

In this work we show how the lacking information can be automatically added to a phrase structure tree to convert it into a dependency tree. Obviously, such conversion cannot be completely accurate because the two representations are (arguably) non-equivalent when it comes to rarer grammatical constructions such as non-projective constructions, e.g.:

In this paper we will ignore such details and will be only concerned with, a bit quick-and-dirty, conversion of the majority of the links. Our main motivation in this work was the application listed above, mainly the study of collocations statistics and its application for syntactic disambiguation and word sense disambiguation.

The paper is organized as follows. Section 2 briefly introduces the corpus that was the base for our experiments. Section 3 presents in detail our transformation procedure and the heuristics we used in the conversion of this specific corpus. Section 4 discusses the experimental results, and Section 5 concludes the paper.

2 The Spanish Cast3LB Treebank

Cast3LB is a corpus of 100 thousand words (approximately 3,700 sentences) created from two corpora: the CLiCTALP corpus (75 thousand words), a balanced and morphologically annotated corpus containing literary, journalistic, scientific, etc. language, and the corpus of the EFE Spanish news agency (25 thousand words) corresponding to year 2000.

The annotation process has been carried out in two steps. In the first step a subset of the corpus has been selected and annotated twice by two different annotators. The results of this double annotation process have been compared and a disagreement typology in sense assignation has been established. After a process of analysis and discussion, a handbook of annota-

(S	clause
(S.F.C.co-CD	clause
(S.F.C	clause
(sn-SUJ	noun phrase
(espec.fp	specifier
(da0fp0 Las el))	determiner *The* $_{\text{feminine plural}}$ *the*
(grup.nom.fp	nominal group
(ncfp000 reservas reserva)	noun *reserves reserve*
(sp	prepositional phrase
(prep	preposition
(sps00 de de))	preposition *of of*
(sn	noun phrase
(grup.nom.co	nominal group
(grup.nom.ms	nominal group
(ncms000 oro oro))	noun *gold gold*
(coord	coordinate
(cc y y))	coordinate *and and*
(grup.nom.fp	nominal group
(ncfp000 divisas divisa)))))	noun *currencies currency*
(sp	prepositional phrase
(prep	preposition
(sps00 de de))	preposition *from from*
(sn	noun phrase
(grup.nom	nominal group
(np00000 Rusia Rusia))))))	noun *Russia Russia*
(gv	verb phrase
(vmis3p0 subieron subir))	verb *raised to_raise*
(sn-CC	noun phrase
(grup.nom	nominal group
(Zm 800_millones_de_dolares	number *800_millions_of_dollars*
800_millones_de_dolares))))	*800_millions_of_dollars*

Figure 1. A sentence with original labels from 3LB Treebank ('The reserves of gold and currency from Russia rose 800 million of dollars').

tion has been produced, where the main criteria to follow in case of ambiguity have been described. In the second step, the rest of the corpus has been annotated following the all words strategy. The lexical items annotated are those words with lexical meaning, i.e., nouns, verbs, and adjectives (Navarro *et al*. 2003).

3 Transformation Procedure

The transformation procedure can be described roughly as follows:

1. Extract the constituency grammar rules from the Cast3LB Treebank;
2. Head Marking: Use heuristics to find the head component of each rule;
3. Recursively use these information of the heads to determinate which rules to find which component will rise up in the tree.

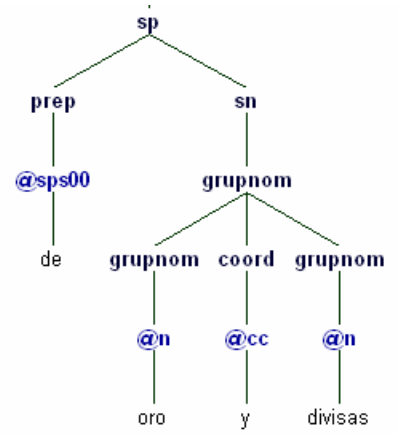
We describe each of these steps below.

3.1 Extracting the Grammar

To extract the grammar from the Cast3LB Treebank we used the following steps:

Simplification of the constituency treebank: The Cast3LB Treebank divides tags in two parts. The first one specifies the part of speech, for example, clause, noun, verb, noun phrase, etc. This for our purposes the most important part of the tag. The second part specifies additional features such as gender and number for noun phrases, or the kind of subordinate clause. These features can be elided to reduce the number of grammar rules without affecting the transformation. For example, for a clause, the Cast3LB Treebank uses: S (clause), S.F.C (coordinate clause), or S.F.C.co-CD (object coordinate clause). We mapped them all to a single label, *S*. For nominal groups, Cast3LB uses grup.nom (nominal group), grup.nom.fp (feminine plural nominal group), grup.nom.ms (masculine singular nominal group), grup.nom.co (coordinate nominal group), etc.; we mapped them to a single label grupnom. Figure 1 shows a part of Cast3LB Treebank using the original labels. Figure 2 shows the same part of Cast3LB.

In order to reduce the number of patterns of the resulting grammar, we also simplified the




```
(sp
  (prep
    (@sps00 de de))
  (sn
    (grupnom
      (grupnom
        (@n oro oro))
      (coord
        (@cc y y))
      (grupnom
        (@n divisas divisa)))))
```

Figure 2. Nodes that only have one leaf marked as heads.

tagging of the Cast3LB Treebank by eliminating all punctuation marks.

Pattern extraction. To extract all the rules of the grammar, each node with more than one child is considered as the left part of a rule, and its children are the right part of the rule. For example, the patterns extracted from Figure 3 are shown in Figure 4. Here *grupnom* is nominal group, *coord* is coordinate, *sp* is prepositional phrase, *prep* is preposition, *sn* is noun phrase, *n* is noun, *espec* is specifier, *S* is clause and *gv* is verb phrase. A clause (S) can be composed by noun phrase (sn), verb phrase (gv) and noun phrase (sn).

3.2 Marking the Head

After extracting all patterns which form the grammar, the head of each pattern is automatically marked using simple heuristics. We denote the dead of a rule with @ symbol. The heuristics we use are as follows:

1. If the rule contains only one element (or only one its element can be a head, see heuristics 10, 11) then it is the head, e.g.:

   grupnom ← @n

2. If the pattern contains one coordinate (*coord*) then it is the head, e.g.:

   grupnom ← grupnom @coord grupnom
   S ← @coord sn gv sn

3. If the pattern contains two or more coordinates, then the first one is the head, e.g.:

   S ← @coord S coord S
   Sp ← @coord sp coord sp

4. If the pattern contains a verb phrase (*gv*) then it us the head, e.g.:

   S ← sn @gv sn
   S ← sadv sn @gv S Fp

5. If the pattern contains a relative pronoun (*relatiu*), then this is the head, e.g.:

   sp ← prep @relatiu
   sn ← @relatiu grupnom

6. If the pattern contains a preposition (*prep*) as its first element followed by only one element whichever it is, then the preposition is the head, e.g.:

   sp ← @prep sn
   sp ← @prep sp

7. If the pattern contains an infinitive verb (*infinitiu*) then it is the head, e.g.:

   S ← @infinitiu S sn
   S ← conj @infinitiu
   S ← neg @infinitiu sa

8. If the pattern contains a present participle (*gerundi*) then it is the head, e.g.:

   S ← @gerundi S

9. If the pattern contains a main verb (*vm*) then it is the head, e.g.:

   gv ← va @vm
   infinitiu ← va @vm

10. If the pattern contains an auxiliary verb (*va*) and any other verb then the auxiliary verb is never the head, e.g.:

    gv ← va @vs

11. If the pattern contains a specifier (*espec*), as its first element, then it is never the head, e.g.:

    sn ← espec @grupnom
    sn ← espec @sp

12. For patterns with noun phrase (*grupnom*) as father node, if the pattern contains a noun (*n*) then it is the head, e.g.:

    grupnom ← s @n sp
    grupnom ← @n sn
    grupnom ← s @n S

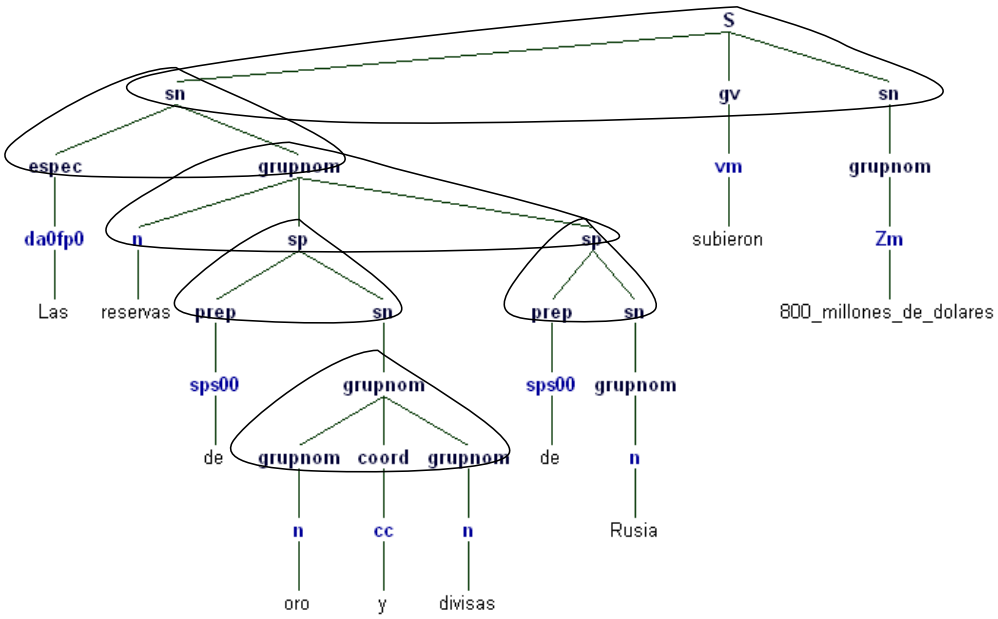


Figure 3. Patterns to be extracted from the sentence 'The reserves of gold and currency from Russia rose in 800 millions of dollars'

grupnom ← grupnom coord grupnom
sp ← prep sn
grupnom ← n sp sp
sn ← espec grupnom
S ← sn gv sn

Figure 4. Extracted patterns of the sentence "The reserves of gold and currency from Russia rose in 800 millions of dollars"

13. For patterns with noun phrase (*grupnom*) as father node, if the pattern contains a noun phrase (*grupnom*), it is the head, e.g.:

    grupnom ← @grupnom s
    grupnom ← @grupnom sn

14. For patterns with specifier (*espec*) as father node, if the pattern contains a definitive article (*da*) then it is the head, e.g.:

    espec ← @da di
    espec ← @da dn

15. If the pattern contains a qualificative adjective (*aq*) and a prepositional phrase (*sp*), then the adjective is the head, e.g.:

    S ← sadv @aq sadv
    sa ← sadv @aq sp sp

If the above heuristics do not allow to unambiguously determine the head, we choose the first element that can be the head (cf. see heuristics 10, 11).

The order of application of the heuristic rules is important. For example, if we apply rule 2 in the pattern: S ← coord sn gv sn Fp, the head would be *gv*, instead of marking the correct head *coord*. For this to occur, Rule 1 should have been applied first.

There are cases for which there is no consensus in the dependency grammar community as to the selection of heads (such as coordination, relative constructions, etc.). We do not attempt here to contribute linguistic arguments in these issues, and the above heuristics reflect just one of possible linguistic options.

3.3 Using the marked heads for the transformation

The transformation algorithm uses recursively the information of the patterns marked with heads to determine which components will rise up in the tree. This means to disconnect the head from its brothers and to place it in the position of father node.

In order to understand more clearly the algorithm, we describe it in detail:

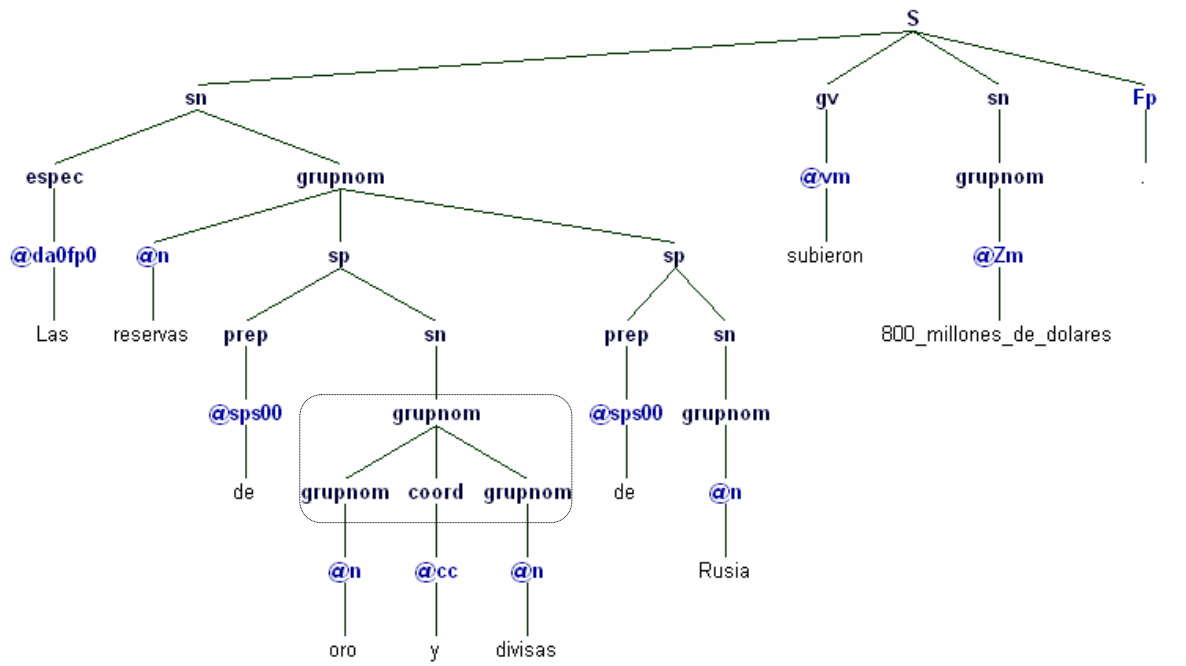


Figure 5. Constituency tree. 'The reserves of gold and currency from Russia raised 800 million of dollars'

1. Traverse the constituency tree in depth from left to right, beginning from the root and visiting the children nodes recursively.
2. For each pattern in the tree, search the rules to find which element is the head.
3. Mark the head in the constituency tree. Disconnect it from its brothers and place it in the position of father node.

The algorithm finishes when a head node is risen up as root. To illustrate an example, consider the following figures.

Figure 5 shows a constituency tree that will be transformed into a dependency tree. Remember that nodes that only have one leaf were marked in the extraction grammar.

Following the algorithm, the first pattern to be found is: grupnom ← grupnom coord grupnom, where *grupnom* is a nominal group and *coord* is a coordinate.

We look in the rules and found that the head of these patterns is the coordinate (coord). We mark the head in the constituency tree and disconnect it by putting it in the position of the father node.

The algorithm follows its execution until the root node is raised. The resulting Dependency tree is shown in Figures 6 and 7.

4 Experimental Results

The algorithm found 2663 grammar rules. From those, 339 (12%) are repeated more than 10

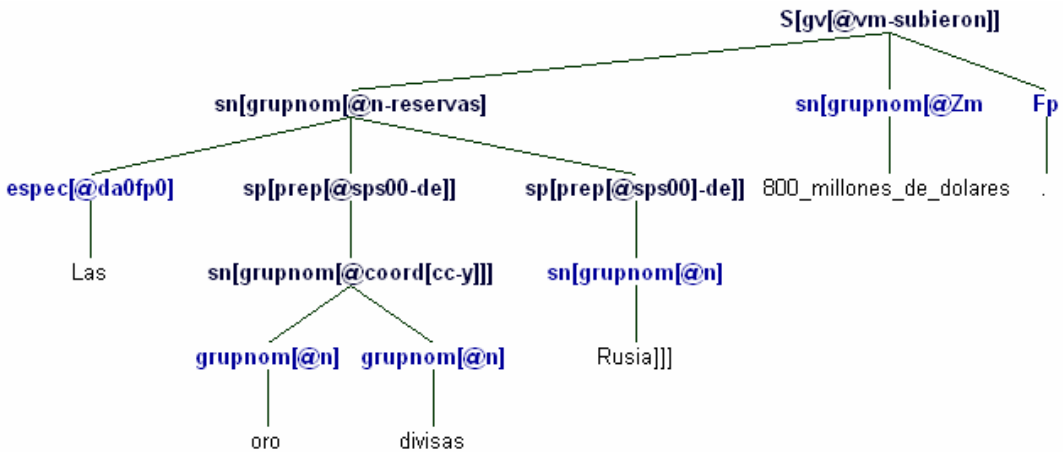


Figure 6. Resulting dependency tree with labels.

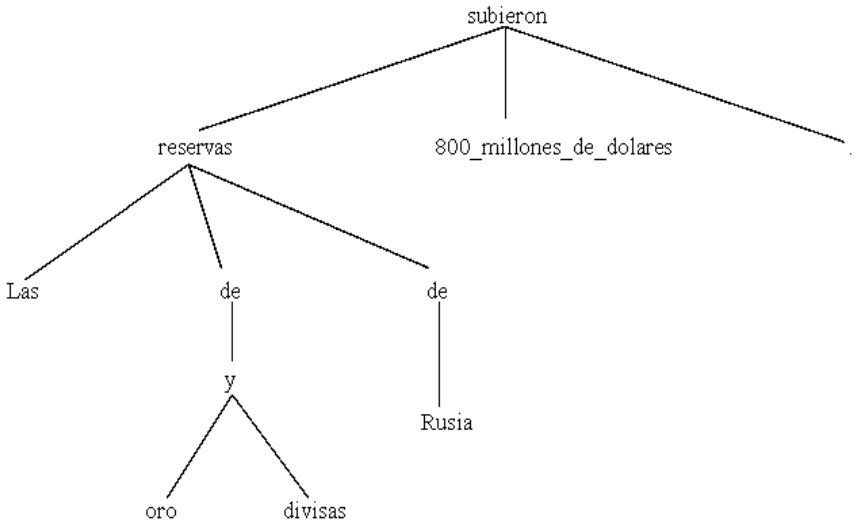


Figure 7. Resulting dependency tree without labels.

times and 2324 (88%) less than 10 times. The twenty most frequent rules with their respective number of occurrences are:

12403	sn ← espec grupnom
11192	sp ← prep sn
3229	grupnom ← n sp
1879	grupnom ← n s
1054	sp ← prep S
968	grupnom ← n S
542	gv ← va vm
535	grupnom ← s n
515	S ← infinitiu sn
454	grupnom ← n s sp
392	grupnom ← n sn
390	grupnom ← grupnom coord grupnom
386	sn ← sn coord sn
368	grupnom ← s n sp
356	gv ← vm infinitiu
343	S ← S coord S Fp
315	S ← S coord S
276	sp ← prep sn Fc
270	grupnom ← n sp sp
268	S ← infinitiu sp

4.1 Identifying Heads in the Rules

The heuristics covered, i.e., thus automatically labeled, 2210 (79.2%) of all extracted grammar rules.

We randomly selected 300 of them and marked them manually. Comparison showed that all but two (99.9%) marks coincided. Figure 8 shows the rules that not matched. These two rules not matched because the heuristic rules do not consider these cases.

Considering the comparison statistics, we believe that at least 95% of the automatically marked rules of Cast3LB are correctly marked.

4.2 Construction of Dependency Trees

We have followed the evaluation scheme proposed by Briscoe *et al.* (2002), who suggest evaluating parsing accuracy based on grammatical relations between lemmatized lexical heads. This scheme is suitable for evaluating dependency parsers and constituency parsers as well, because it considers relations in a tree which are present in both formalisms, for example [Det *car the*] and [DirectObject *drop it*]. For evaluation we extract triples from the dependency trees found by our method and compare it with manually extracted triples from the same Cast3LB treebank.

A triplet is a dependency relation between a father node with a children node and the type of their relation. For example, the dependency triplets extracted from the phrase *The old man loves the young lady* are:

love SUBJ man	love OBJ lady
man DET the	lady DET the
man ADJ old	lady ADJ young

The algorithm extracted 65,997 dependency

Automatically marked	Manually marked
infinitiu <-- van0000 vmp00sm sps00 @infinitiu	infinitiu <-- van0000 @vmp00sm sps00 infinitiu
S.F.C.co-CD <-- conj.subord S.F.C @coord S.F.C	S.F.C.co-CD <-- @conj.subord S.F.C coord S.F.C

Figure 8. Rules that not matched.

triples from the whole Cast3LB treebank.

For evaluation, we randomly selected 35 sentences from the treebank and manually converted them to dependency trees, which gave 399 dependency triples. Then we applied our procedure to these sentences. Since for a sentence of *n* words there must be (n – 1) triplets, our procedure also output 399 triplets, of them, 368 (92%) coinciding with those manually identified. Extrapolating this statistics, we infer that more than 90% (some 60,000) of the dependency triples that we extracted from Cast3LB Treebank are correct.

5 Conclusions

Dependency representation of syntactic structure has important advantages in certain applications, such as nearly everything related to lexicalization and lexicography. However, the majority (though not all) of existing tools and resources, such as parsers, grammars, and treebanks, are oriented to constituency approach.

We have presented a simple unsupervised technique that allows automatically transforming constituency trees into dependency trees. The technique uses certain simple heuristics that depend on the specific tagset used in the given treebank or grammar. Our technique does not deal with difficult or arguable phenomena in dependency syntax, but still recovers the bulk of dependency relations. Such a bit quick-and-dirty results are quite usable in most practical applications.

This allows for reuse of existing parsers or treebanks for the applications that require dependency structures.

Acknowledgements

The work was done under partial support of Mexican Government (SNI, CONACyT, CGPI-IPN, COFAA-IPN, PIFI-IPN). Cast3LB is part of the 3LB project financed by the Science and Technology Ministry of Spain, 3LB, (FIT-150500-2002-244 and FIT 150500-2003-411). We thank Jordi Atserias for useful discussions and help.

References

Bolshakov, Igor A. A Method of Linguistic Steganography Based on Collocationally-Verified Synonymy. *Information Hiding 2004*, *Lecture Notes in Computer Science*, 3200 Springer-Verlag, 2004, pp. 180–191.

Bolshakov, Igor A., Alexander Gelbukh. A Large Database of Collocations and Semantic References: Interlingual Applications. *International J. of Translation*, V.13, No.1–2, 2001, pp. 167–187.

Bolshakov, Igor A., Alexander Gelbukh. On Detection of Malapropisms by Multistage Collocation Testing. *NLDB-2003, 8th Int. Conf. on Application of Natural Language to Information Systems*. Bonner Köllen Verlag, 2003, pp. 28–41.

Briscoe, Ted, John Carroll, Jonathan Graham and Ann Copestake. 2002. Relational evaluation schemes. In: *Proceedings of the Beyond PARSEVAL Workshop at the 3rd International Conf. on Language Resources and Evaluation,* Las Palmas, Gran Canaria, 4–8.

Gelbukh, Alexander. Syntactic disambiguation with weighted extended subcategorization frames. *Proc. PACLING-99*, 1999, pp. 244–249.

Mel'čuk, Igor A. *Dependency Syntax: Theory and Practice*. State University Press of New York, 1988.

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

Sag, Ivan, Tom Wasow, and Emily M. Bender. *Syntactic Theory. A Formal Introduction* (2nd Edition). CSLI Publications, Stanford, CA, 2003.

Sowa, John F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley Publishing Co., Reading, MA, 1984.

Yuret, Deniz. *Discovery of Linguistic Relations Using Lexical Attraction*, PhD thesis, MIT, 1998.