Enhancing Dependency Analysis by Combining Specific Dependency Parsers

Miguel Ballesteros, Jesús Herrera, Virginia Francisco, Pablo Gervás
miballes@fdi.ucm.es, jesus.herrera@fdi.ucm.es, virginia@fdi.ucm.es, pgervas@sip.ucm.es.
Department of Software Engineering and Artificial Intelligence,
Universidad Complutense de Madrid, Spain

Abstract

Nowadays syntactic dependency parsers show very good performance, which is usually around 85% for Labeled Attachment Score (LAS). In the present work, we studied the combination of different small dependency parsers after an in–depth study of the errors produced by existing parsers at the word level. Each one of these small parsers is able to tackle one single kind of dependency, so its parsing becomes at the word level. These specific parsers are extremely accurate when parsing concrete words, therefore their outputs can be used to replace wrong sections of the outputs given by general purpose parsers. The combined action of all them leads to a more accurate parsing.

1 Introduction

Statistical Dependency Parsing is one of the most important topics on Natural Language Processing (NLP). In the 10th edition of the Conference of Computational Natural Language Learning (CoNLL) a first shared task on Multilingual Dependency Parsing, namely the CoNLL-X Shared Task, was performed (Buchholz & Marsi, 2006a). Thirteen different languages were involved and parsing accuracy was studied. In this Shared Task, each participant implemented a parsing system that could be trained to parse all those languages.

In spite of the overall high performance of the systems presented to the CoNLL–X Shared Task, the highest accuracy achieved was 90.71% Labeled Attachment Score (LAS) for Japanese. Even for Japanese, an improvement of more than 9% was possible. In addition, any given parser performed more or less accurately, depending on the language processed, and some languages were parsed with relatively low accuracy, giving the impression that current technologies are not enough good for them. A good example from the Shared Task is Turkish, whose best parser performed 65.68% LAS, while for the other languages, except for Arabic, at least one system achieved 72% LAS. These facts led us to think about the connection between training corpora and parsing accuracies.

Besides these considerations, we have evidences that the size of the training corpus is not necessarily linked to parsing accuracy. For instance, in CoNLL–2007 MaltParser (Nivre et al., 2007a) reached better results for Italian (84.4% LAS), by using a relatively small training corpus of approximately 71,000 wordforms, than for Czech (77.98% LAS), with a training corpus of approximately 432,000 wordforms. Not only MaltParser but all the other systems participating reached worse results for Czech than for Italian in that Task. So, on the one hand we have that parsing performance is not the same for every language processed by a system and, on the other hand, nothing guarantees that we will obtain better parsings by increasing the size of the training corpus. At this point, given that we can not internally modify these systems, *what can we do to make them do their best?*

Once stated the unknown factors we wanted to solve, we determined the tool that would make possible our research: MaltParser. This is the publicly available software that is contemporaneous to the system presented by Nivre's group in the CoNLL–X Shared Task, in which Spanish was proposed for parsing and Nivre's group achieved very good results. Since Spanish is our mother tongue and we had experience with MaltParser, we developed the present work focusing on Spanish dependency parsing using MaltParser (the eager standard version). Nevertheless, the techniques presented here can be easily extended to other languages and/or parsers.

Then, with this work we were willing to improve not only current Malparser's overall results for Spanish, but to reach a better complete–match accuracy[1] as well. Thus, the end user of a dependency parser receives a more satisfactory solution. To do this, we tried to find answers to the following questions:

1. Is it viable to improve accuracy by combining the action of small and specific parsers?

2. Which actions to improve accuracy can be automated?

There are some work published that inspired our proposal, such as (McDonald & Nivre, 2007) and (McDonald & Nivre, 2011). In them, they propose the combination of two data–driven dependency parsers (MaltParser and MSTParser) by using stacking at training time. This led us to think about a hybrid parsing combination system, but not a classic one. Usually, in such a system every individual component (version) computes the whole output and a further voting decides which one will be the final output. Nevertheless, our *n–version* proposal is as follows: each component computes a part of the whole output and then all these parts are conveniently combined to get the final output. So the small specific parsers compute sections of the final dependency tree and, at parsing time, they are joined together to built the whole tree.

In the following sections we describe the development of our work. But, first of all, in Section 2, we show relevant related work: the CoNLL–X Shared Task, the corpus that was used in it for Spanish (the AnCora corpus, named Cast-3lb at this time), and an introduction to statistical dependency parsing and hybrid systems, including comments about MaltParser and evaluation measures. Once the basis for our work is stated we continue by describing in Section 3 the low level techniques that we designed to study the feasibility of our n-version proposal. Finally, Section 4 presents our conclusions and suggestions for future work.

2 Related Work: Dependency Parsing

The basic idea of Syntactic Dependency Parsing is that syntactic structure consists of lexical items linked by binary asymmetric relations called dependencies, and we can build a dependency structure for a sentence which is a labeled directed tree, consisting of a set of nodes, labeled with words, and a set of arcs labeled with dependency types (Nivre, 2006).

There has been a surge of interest in dependency parsing, motivated both by the efficiency and the potential usefulness of bi-lexical relations that allow us to consider tasks that would be hard to process with a different kind of syntactic parsing, such as phrase structure parsing. For instance, recent applications such as textual entailment recognition (Herrera et al., 2005), relation extraction (Culotta & Sorensen, 2004), machine translation (Ding & Palmer, 2005), question–answering systems (Cui et al., 2005), negation scope classification (Councill et al.,

[1]Nowadays dependency parsers are usually measured using token–based measures, but we consider also important to measure the percentage of sentences in the test set with correct labeled graph known as *complete match accuracy* (Yamada & Matsumoto, 2003).

2010), speculation scope classification (Ballesteros et al., 2011), synonym generation (Shinyama et al., 2002), lexical resource augmentation (Snow et al., 2005) or sentence simplification (Ballesteros et al., 2010a) used dependency parsing as a basis for their work. This success of dependency parsing motivated us.

In the following Subsections we show the state of the art in Statistical Dependency Parsing, different Hybrid approaches, the Spanish AnCora corpus and the parser selected for our proposal: MaltParser.

2.1 Statistical Dependency Parsing

Our work is focused on improving Statistical Dependency Parsing, i.e., systems trained with a set of examples able to return dependency parsings of free text. Nowadays, there are a wide range of corpora derived from the CoNLL–X Shared Task (Buchholz & Marsi, 2006a) and the CoNLL 2007 Shared Task (Nivre et al., 2007a). So the work shown in this paper can be expendable to other languages using the corpora available. In the case of Spanish parsing the freely available treebank is AnCora, see subsection 2.3.

Nowadays there are a pretty wide range of approaches to the resolution of the Dependency Parsing problem. Rule–based parsers were one of the most significant ones, having Minipar (Lin, 1998) as a classic reference. These kinds of parsers are very language–dependent, needing the development of a complete set of rules for every language to be parsed. This is why, in the last years machine learning–based parsers have become the most popular ones, as shown in (Buchholz & Marsi, 2006b). It permits to have a parser for a new language by training a model able to parse it, avoiding the programming of a whole new parser. But, on the other hand, annotated corpora for training are needed. These parsers were established after the celebration of the CoNLL Shared Tasks on Dependency Parsing in two type of approaches (but there are also others), described in (McDonald & Nivre, 2011), and have become the most implemented ones. They are the following:

- The Graph–based model (Eisner, 1996), (McDonald et al., 2005), that parameterizes parsing models by dependency arcs and tries to predict entire trees by using the input given besides spanning tree algorithms. The best current example for this technology is MSTParser, but we have also some other similar systems such as Corton's (Corston-Oliver & Aue, 2006), Dreyer's (Dreyer et al., 2006) or more recently Bohnet's (Bohnet, 2010).

- The Transition–based model (Nivre et al., 2004), (Yamada & Matsumoto, 2003), in which parsing models are parametrized by state transitions and the system learns to predict new trees by means of greedy inference, given the input and a wide history of examples. MaltParser is the most representative parser for this technology. Johansson's (Johansson & Nugues, 2006), Cheng's (Cheng et al., 2006) and Wu's (Wu et al., 2006) systems follow derivations of this model as well.

2.2 Hybrid Methods for Parsing Combination

Despite an active research community is devoted to dependency parsing, few works on improving parsing accuracy without modifying the entire machine learning–based parsers, but combining several of them have been developed. They can be divided in parser combination by stacking, parser combination by voting and parser combination by dual decomposition. Here we briefly describe the most relevant ones related to our work.

Joakim Nivre and Ryan McDonald used stacking to integrate graph–based and transition–based dependency parsers (McDonald & Nivre, 2011). They showed how to integrate both parsers at training time using stacking, so that the two complementary models can learn from one another. This work inspired us to build the n–version parser, described in Section 3. Also, we took their study on the lengths of sentences as the starting point for our first work to study dependency parsing accuracy. Moreover, they also studied the progress that has been made through dependency analysis by means of a deep error analysis. Therefore, they combined and compared the two dominant approaches (transition–based models and graph–based models) towards a better accuracy.

We considered the work done by (Zeman & Zabokrtský, 2005) as well. They proposed an approach that combines several dependency parsers for Czech. Their goal was to tell, for each word, which parser is the most likely to pick its dependency correctly, by switching the information with a voting system. This work is similar to our n–version dependency parser proposal, but it differs because we only use a single system, trained n times, while they use seven different parsers.

Moreover (Sagae & Lavie, 2006), used spanning tree parsing to do ensemble parsing and (Koo et al., 2010) used dual decomposition to combine the Chu–Liu Edmonds algorithm[2] with a third–order dynamic programming algorithm. This approach is far from our proposal.

Another relevant work on hybrid parsing is (Chen et al., 2009), which shows an approach in which English and Chinese parsing results are enhanced by means of subtrees from auto–parsed data; then new subtree–based features for parsing algorithms is constructed. This work is also related to ours. They studied prepositions as function words and how to coordinate conjunctions, as we show in the present work. Also, we considered their way of using subtrees and their idea of tackling complete–match accuracy loss.

2.3 The AnCora Corpus

AnCora (Palomar et al., 2004; Taulé et al., 2008) is a corpus composed of 95,028 wordforms and 3,512 sentences that contain sentences from different domains annotated with their dependency analyses in Spanish. AnCora was developed by the Clic group (University of Barcelona). This corpus was built automatically by using the Cast3lb constituency treebank (Civit et al., 2006), with a linguistic validation step. AnCora was arranged according to the CoNLL[3] data format because it was provided as a training and test corpus for Spanish dependency parsing in the CoNLL–X Shared Task.

This corpus was divided into two fractions for the CoNLL–X Shared Task. One of them was provided as a training set for Spanish (89,334 wordforms) and the other one was used as test set (5,694 wordforms). We used the same version of AnCora provided in the CoNLL–X Shared task.

2.4 MaltParser

MaltParser (Nivre et al., 2006; Nivre et al., 2007b) is a training–based system in which by using a corpus like AnCora it is possible to train a system capable of parsing a sentence with dependency relations. It implements the transition–based approach to dependency parsing, which is basically a nondeterministic transition system for mapping sentences to dependency trees and a classifier that predicts the next transition for every possible system configuration. For training, MaltParser uses support vector machines (Cortes & Vapnik, 1995) and it consists of a single classification decision (*create arc, shift reduce, etc.*) for each node. MaltParser offers a wide range of parameters for optimization, including nine different parsing algorithms, an expressive specification language that can be used to define arbitrarily rich feature models, and two different machine learning libraries.

Therefore, MaltParser uses history–based feature models to predict the next action in the construction of a dependency structure, which means that it uses features of the partially built dependency structure and features of the annotated input string. More precisely, features could include the wordform (LEX), part–of–speech (POS) or dependency type (DEP) of a token defined relative to one of the data structures STACK, INPUT and CONTEXT. A feature model is defined in an external feature specification[4] and it allows feature models of arbitrary complexity.

Finally, it is worth to mention that with MaltParser it is possible to perform parsing in linear time for projective dependency trees and in quadratic time for arbitrary trees (non-projective) (Bosco et al., 2010).

[2]http://www.softpanorama.org/Algorithms/Digraphs/mst.shtml

[3]`http://ilk.uvt.nl/conll/`

[4]An in–depth description of these feature models can be found in `http://maltparser.org/userguide.html`

In our work, the first step was to replicate the participation of Nivre's group in the CoNLL–X Shared Task for Spanish. We trained MaltParser with the section of AnCora that was provided as training corpus in the CoNLL–X Shared Task (89,334 wordforms) and the system was set as referred by Nivre's group in (Nivre et al., 2006). These settings are used in all the experiments shown in this paper. Once a model was obtained, we used it to parse the section of AnCora that was provided as a test set in the CoNLL–X Shared Task (5,694 wordforms). We obtained more or less the same results to Nivre's group in the Shared Task (considering that MaltParser has evolved from 2006 to the present), i.e., 81.87% LAS, 85.24% UAS and 90.24% LA. These results serve us as a baseline for our work, which is presented in the following Section.

3 Hybrid Combination: An N–Version Dependency Parser

An end user could usually expect a high complete–match parsing accuracy (at the sentence level) rather than a high overall parsing accuracy. But nowadays a remarkable percentage of sentences in Spanish show at least one error when parsed by MaltParser. Our hypothesis is that by enhancing complete–match accuracy not only overall accuracy will be enhanced but end-user satisfaction will be increased.

In this Section we discuss our proposal on combining different dependency parsers (N parsers), all of them integrated conform a complete dependency parser. Therefore, we called the final system N–Version Dependency Parser.

Section 3.1 analyze complete–match accuracy, Section 3.2 shows a set of experiments carried out to confirm our hypothesis, Section 3.3 shows an algorithm developed towards an N–version dependency parser.

3.1 Complete–Match Parsing Accuracy

Despite a high overall parsing accuracy only 358 wordforms of the test corpus obtain a 100% LAS, UAS and LA in all parsed sentences, i.e., only 6.3% of the wordforms. When considering sentences, only 38 sentences of the test corpus (18.4% of them) were parsed without errors. That is, a 18.4% LCM (Labeled Complete Match) when setting the system with the same specifications of the CoNLL–X Shared Task.

We found that there is a small set of words that normally show an incorrect attachment, labeling or both. These words are the prepositions "a" (*to*), "de" (*of*), " en" (*in*), "con" (*with*), "por" (*for*), the conjunction *and* (which has two wordings: "y" or "e"), and the nexus "que" (*that*). They are not only the most frequent source of mistakes, they also produce more mistakes than other words with the same frequency. All these words sometimes cause errors in the dependency, in the head tag, or in both tags. For instance, there are only 20 sentences (340 wordforms) in the test corpus with only one error after parsing. That is, 9.7% of the corpus' sentences and 5.98% of its wordforms. We found that in 10 of these 20 sentences the only failure is caused by one of the words listed above. The misanalysis of these words could be the reason that the resultant tree might be less useful, because these words (such as prepositions, conjunction and nexus) are function words and in many cases the root of subtrees. Therefore, our hypothesis consists in the improvement of the analysis of these specific words towards a better complete–match accuracy and also a better token–based accuracy (global measures, such as Labeled Attachment Score).

3.2 Our Proposal: Obtaining N Dependency Parsers

Our idea was to determine if these "difficult" words (shown above) could successfully be parsed by specific parsers while a general parser would parse the non–troubled wordforms. Therefore, the N–version dependency parser works as follows: the general dependency parser parses the whole sentence. If a pattern is detected, the specific dependency parser associated to that pattern parses the sentence in the same way that the general parser did. Finally, a program would swap the node obtained for the specific word, and the rest of the dependency-parsed tree given by the specific dependency parser will be ignored. In this way, we are obtaining a better dependency-parsed tree for the sentence.

Therefore, we did an in–depth study of each one of the words listed in the previous subsection. This study, as described in (Ballesteros et al., 2010b), consisted of finding out the set of different cases in which each word could be attached and labeled, and training a specific parser for each case found. As a result, we found that the conjunction is the word that caused a parsing error more frequently. This is why we selected the conjunction as a first case to study in order to determine if low level techniques are feasible to improve parsing accuracy. To this end we developed a manual analysis of the Corpus AnCora. We extracted from the corpus every sentence containing a conjunction ("y" or "e" in Spanish) conforming a total amount of 1,586 sentences with at least one conjunction. We inspected these sentences to find labeling patterns. This way we obtained a list of patterns that depend on the conjunction's action. For instance, a pattern is given when conjunction acts as a nexus in a coordinated copulative sentence, and another pattern is given when it acts as the last nexus in a list of nouns. For instance, in the following sentence: *Los activos en divisas en poder del Banco Central* **y** *el Ministerio de Finanzas se calculan en dólares estadounidenses* **y** *su valor depende del cambio oficial rublo–dólar que establece el Banco Central (The foreign exchange assets held by the Central Bank and the Ministry of Finance are calculated in U.S. dollars and its value depends on the official ruble–dollar exchange rate established by the Central Bank)* the first y is a nexus between the proper nouns *Banco Central (Central Bank)* and *Ministerio de Finanzas (Ministry of Finance)* and the second *y* acts as a coordinated copulative nexus. These patterns guided the approaches described below. Moreover we developed a similar process for the rest of function words that are listed in Section 3.1, the results of this study is shown in Section 3.2.2.

The first specific parser that we tried to obtain was supposed to parse quoted sentence sections containing conjunctions, shown in Subsection 3.2.1. This situation is quite common and corresponds to one of the labeling patterns that we have identified as problematic. In Subsection 3.2.2 we show the whole approach, which consists of an in–depth study about all the words that are more frequently incorrectly parsed, in the way of studying the feasibility of this idea. The study consists of finding the different ways that these words are attached and labeled, and training a specific model for each case, and to build automatically training corpora for each case.

3.2.1 First Approach to an N–Version Dependency Parser

Our first attempt was a specific model for coordinated copulative sentences to check the feasibility of building n specific parsers in the way of achieving higher accuracy. To this end, we built a specific training corpus with the set of unambiguous coordinated copulative sentences contained in the section of AnCora that was provided as training corpus in the CoNLL–X Shared Task. This specific training corpus contains 361 sentences (10,561 wordforms). Then, we parsed all the coordinated copulative sentences contained in the section of AnCora that was provided as test corpus in the CoNLL–X Shared Task (16 sentences, 549 wordforms). We set the experiments described above with the same feature model (see the description of the feature model in Section 2.4), that Nivre's group used in its participation in the CoNLL–X Shared Task. We found that the conjunction was incorrectly parsed 8 times (in a test set containing 16 conjunctions). This fact led us to investigate with different feature models. After a few failed attempts we found a feature model where 12 of the 16 conjunctions were parsed correctly. Despite the results being enhanced by using the new feature model, the general parsing model (configured as established in the CoNLL–X Shared Task) parses 13 of these 16 conjunctions without errors. It could mean that specific models are not feasible for our objectives.

3.2.2 Definitive Approach: N Parsers

Since the accuracies reached by both models in the first approach were very similar, we developed some other experiments to confirm or reject our hypothesis. Thus, we tried new specific parsers for other combinations of all the function words that we took into account. We set a parser for each specific pattern found for each specific word. Once the sentence is parsed with the specific model, the result for the "problematic" word is replaced in the resulting tree obtained by the general model. The labeling given to each word by the specific parser is cut from this parsing and pasted into the parsing given by the general model, by replacing the labeling given to these words by the general parser. This easy solution is possible because these words can be changed without affecting the rest of the parsing, because they are function words that are normally the root of different subtrees.

The results obtained for all these words are shown in Table 1. They are usually better when using a specific parser than when using the general parser. But sometimes the specific parsers get the same accuracy as the general parser, so it does not make sense to use the specific parser in such cases. For instance, when parsing the word *de* when attached to an adjective or an adverb, both the general parser and the specific parser show 100% LAS. Only when the word *y* (or *e*) acts as a nexus in coordinated copulative sentences we could not find a specific parser better than the general parser (the general parser reaches 81.3% $\text{LAS}_{y/e}$ and the specific parser reaches 75% $\text{LAS}_{y/e}$). In 21 of the 28 cases identified it is better to use the specific parsers. Further research could arrive at better results for the specific parsers that do not reach 100% LAS yet.

Afterwards, we recomputed LAS, UAS and LA for this combined parsing, which are better than those obtained with the general parsers, as shown in Table 1. It means a slight enhancement with respect to the results given by the general parsing model. In addition, in the combined parsing these words do not belong to the set of words that are most frequently incorrectly parsed. This improvement seems to indicate that this n–version parsing model is feasible and overall accuracy could be substantially improved.

In some cases the given improvement seems spectacular. For instance, when parsing the word *de* when attached to a verb, the general parser shows 0% LAS and the specific parsers show 100% LAS. It is due to the small amount of samples present in the test corpus. For instance, if the test set contains only one sample for a specific case and this sample is correctly parsed, then we obtain 100% LAS. But it does not mean that the parser will parse every given sample of this case with 100% LAS. For the given example the test corpus contained only 4 samples. All these samples were wrongly parsed by the general parser but perfectly parsed by the two specific parsers involved. Therefore LAS was enhanced from 0% to 100%, but this is for the given test corpus. If the test corpus contained more samples, perhaps the specific parsers might not have reached 100% LAS. Usually the local improvement obtained by the specific parsers is very high, but as said before it must be taken cautiously due to the small amount of samples in our test corpus, which are usually between 2 and 10 for each case, 30 being the maximum. Nevertheless, parsing accuracy is reasonably homogeneous and similar accuracies should be expected even when increasing the number of samples in the test set.

In addition, we found that the word *de* attached to a verb with the undefined label "–" is a given case in the training corpus that is not given in the test corpus. Of course, for this situation no error is given by the general parser, but how can we know if the parser can tackle such a case if it is not present in the test corpus? This is why, if we want to obtain a high performing parser, we must carefully rebuild the train and test corpora.

3.2.3 Summary of the N–Version Dependency Parser Approaches

As seen in previous subsections, as a result of the use of specific parsers, complete–match accuracy can be improved. And this redounds to the improvement of the overall accuracy. Dependency parsers can be useful for human end-users, who would presumably use such parsers to analyze little pieces of text. So end-users would feed dependency parsers with isolated sentences. In this case, a single error in the parsing of one sentence should not be desirable. This is because the developers of dependency parsers should care about high complete–match accuracy.

Table 1: Attachment and labeling for all the studied words in AnCora. Specific LAS for each word and case, before and after the application of our method. The left arrow ($^{\leftarrow}$) after a part of speech indicates that this part of speech is before the considered word in the sentence. The right arrow ($^{\rightarrow}$) indicates that the part of speech is after the word.

		Case					
Word		#1	#2	#3	#4	#5	#6
y/e	**Label**	–	–	–	–		
	Attached to a	verb$^{\leftarrow}$	proper noun$^{\leftarrow}$	common noun$^{\leftarrow}$	adjective$^{\leftarrow}$		
	LAS$_{y/e}$ before	***81.3%***	80%	66.7%	80%		
	LAS$_{y/e}$ after	75%	***100%***	***80%***	***100%***		
a	**Label**	CD	CI	CC	CREG	–	–
	Attached to a	verb$^{\leftarrow}$					noun$^{\leftarrow}$
	LAS$_{a}$ before	62.5%	42.9%	60%	25%	***0%***	50%
	LAS$_{a}$ after	***87.5%***	***100%***	***100%***	***75%***	***0%***	***100%***
de	**Label**	CC	CREG	–	–		
	Attached to a	verb$^{\leftarrow}$		adverb$^{\leftarrow}$ adjective$^{\leftarrow}$	noun$^{\leftarrow}$		
	LAS$_{de}$ before	0%	0%	***100%***	83.3%		
	LAS$_{de}$ after	***100%***	***100%***	***100%***	***96.7%***		
que	**Label**	SUJ	–	SUJ			
	Attached to a	verb$^{\rightarrow}$		verb$^{\leftarrow}$			
	LAS$_{que}$ before	88.5%	86.4%	0%			
	LAS$_{que}$ after	***92.3%***	***95.5%***	***100%***			
en	**Label**	CC	CC	CREG	–		
	Attached to a	verb$^{\rightarrow}$	verb$^{\leftarrow}$		noun$^{\leftarrow}$		
	LAS$_{en}$ before	***83.3%***	92.6%	50%	62.5%		
	LAS$_{en}$ after	***83.3%***	***100%***	***100%***	***87.5%***		
con	**Label**	CC	CREG	–	–		
	Attached to a	verb$^{\leftarrow}$			noun$^{\leftarrow}$		
	LAS$_{con}$ before	60%	40%	***100%***	66.7%		
	LAS$_{con}$ after	***80%***	***100%***	***100%***	***83.3%***		
por	**Label**	–	CAG	CAG			
	Attached to a	noun$^{\leftarrow}$	comma$^{\leftarrow}$	adjective$^{\leftarrow}$			
	LAS$_{por}$ before	***100%***	***100%***	80%			
	LAS$_{por}$ after	***100%***	***100%***	***100%***			

After parsing the test corpus with our n–version parser we found that 42 (20.3% LCM) of the parsed sentences showed no parsing errors, while 38 (18.4% LCM) of them where perfectly parsed with the general parser. This improvement of the complete–match accuracy, as shown in (Ballesteros et al., 2010b), has consequentially, not only a better experience for human end-users but also an improvement of overall accuracy. When parsing the test corpus by combining the action of the general parser and our proposed specific parsers, we obtained the following results for overall accuracy: 82.68% LAS, 85.73% UAS and 90.84% LA. This means an improvement of 1.38%

LAS, 1.06% UAS and 0.78% LA in overall accuracy with respect to the results of the general parser alone.

N–version parsers are a way to improve parsing accuracy by systematically avoiding the errors given by a general parser. Nevertheless our approaches show discrete improvement. This improvement is greater when eliminating the errors caused by a frequent word, as shown in Figure 1. Each set of bars shows the increments of LAS, UAS and LA when cumulatively adding the action of specific parsers for each word considered. The first word for which we added the action of its specific parsers was the conjunction (*y* o *e*), because the conjunction is the word most frequently parsed wrongly by the general parser. Following this idea, we cumulatively added the action of specific parsers for each of the words considered, starting with those that caused most parsing errors when using the general parser. In the end, when adding the action of specific parsers for the word *por*, we obtained the action in synergy for all the specific parsers listed in Table 1 and the general parser. We can observe in Figure 1 that LAS, UAS and LA increased notably when adding the action of specific parsers for the conjunction (*y/e*) and the preposition *a*. In fact LA did not increase when the action of a specific parser for the conjunction was added, but this is because the general parser does not fail when attaching the conjunction (only with labeling).

In general terms the more infrequent the word that causes parsing errors is, the less the contribution of its specific parsers to the overall action is. So the effort of building specific parsers may not be worthwhile, given the improvement obtained. It is of interest to remark that the conjunction causes 56 parsing errors with the general parser, *a* causes 48 errors, *de* 44 errors, *que* 42 errors, *en* 37 errors, *con* 17 errors and *por* 16 errors. Also, the increments obtained are not regular and this is not only because of the number of samples of each considered case that exist in the test corpus but also the accuracy of their specific parsers.

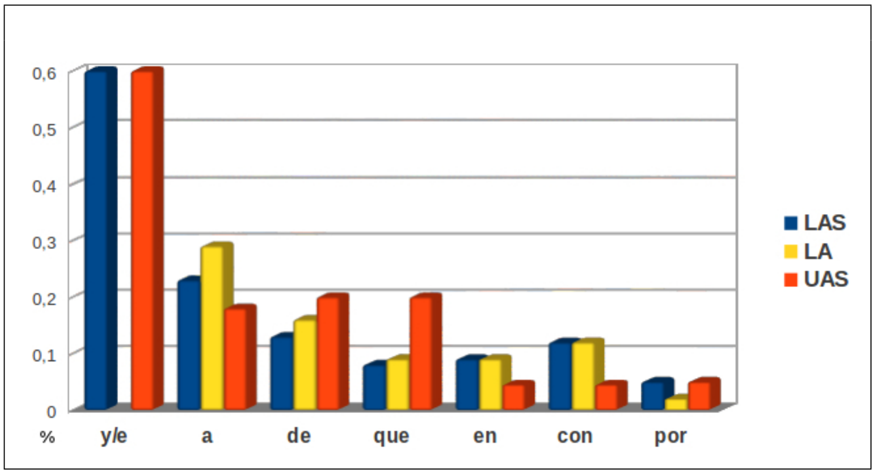

Figure 1: Increments of overall LAS, UAS and LA due to the action of specific parsers that avoid the most frequent errors, given by the specific words.

Although a large percentage of the most frequent errors given when parsing with the general parser are eliminated with the n–version parser, a remarkable amount of errors remains. After our efforts, a 17,32% improvement in LAS is still required to reach a perfect parsing. Since specific parsers have been developed for only a small set of words, some other words remain without a specific treatment and produce errors. This means that a great effort is needed to reach a perfect parsing. A lot of errors could be avoided by implementing more complex n–version parsers, covering a bigger amount of "difficult" words than the ones presented here. But some other errors could be inherent to the implementation of the parsing algorithm or derived from inconsistencies of the corpus and cannot be avoided. Also, as suggested in (Ballesteros et al., 2010c) and in Section 3.2.2, some other errors could be treated by carefully building the training corpora.

3.3 The N–Version Dependency Parser Algorithm

Once we concluded that the combination of several specific parsers could be a feasible technique to enhance parsing accuracy, the following step was to develop it automatically, that is, the algorithm that makes all the specific parsers work in synergy.

Our approach for the algorithm that sends each different wordform to the most appropriate specific parser is based on pattern matching and rules. So when a certain pattern is recognized in the sentence to be parsed, it is sent to the most suitable specific parser by means of a rule. For the time being, this algorithm has been implemented only for the preposition “a” and for the conjunction, which in Spanish has two wordings: “e” and “y”, because these two are the words most frequently parsed incorrectly by the general parser. It works as follows:

1. A sentence is parsed with the general parser. In Figure 2 we show how the general parser parses the sentence: *Trasladó el material a Madrid* [*he (or she) moved the material to Madrid*], we can observe that the general parser makes an error in the node containing the preposition “a”. The correct attachment for this node must be the main action of the sentence: *Trasladó*, and the correct label is “CC” that is the adjunct of the verb. The general parser produces an error in both things.

2. If the algorithm detects that there is a conjunction or a preposition “a” in the sentence, it sends the whole sentence to the most appropriate specific parser. Each parser is set with different specifications, shown in Section 3.2.2.

3. The selected specific parser parses the sentence. In Figure 2 we can also observe how the specific parser parses the same sentence as the general parser but does not make the same error as the general parser. In this case, the specific parser correctly parses the node containing the preposition “a” (nevertheless it does not correctly parse other sections of the sentence).

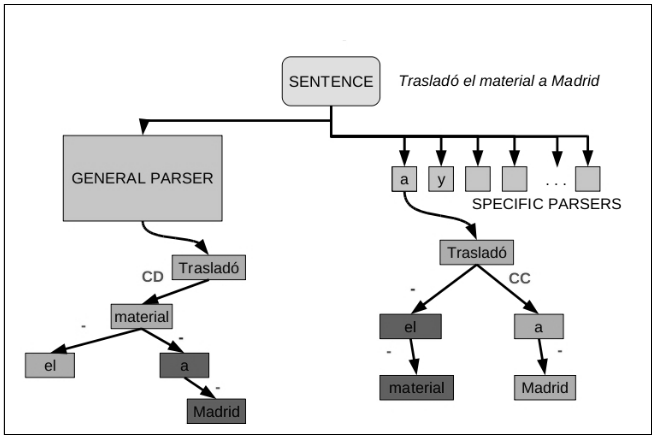

Figure 2: The Specific Parser parsing the sentence: *Trasladó el material a Madrid* [*he (or she) moved the material to Madrid*].

4. The algorithm removes the node containing the conjunction or the preposition “a” from the tree returned by the general parser. In Figure 3 we can observe how the system avoids the useless information given by both the general and the specific parser.

5. The algorithm inserts the node containing the conjunction or the preposition “a”, produced by the specific parser into the general parsed tree. In Figure 4 we can observe how the algorithm inserts the node containing

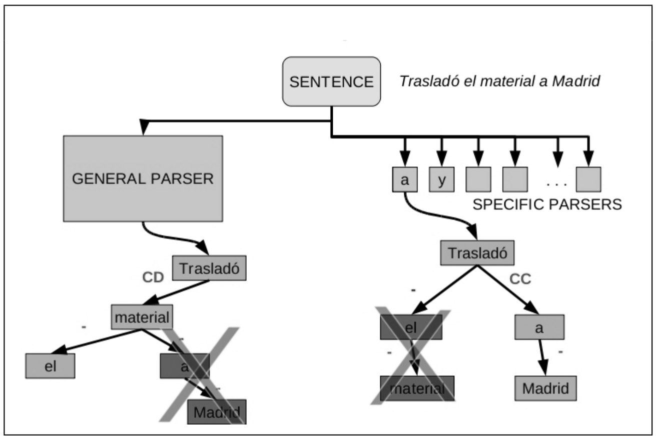

Figure 3: Removing useless information given by both parsers.

the preposition "a" into the tree given by the general parser. When inserting the node, the algorithm also inserts its subtree.

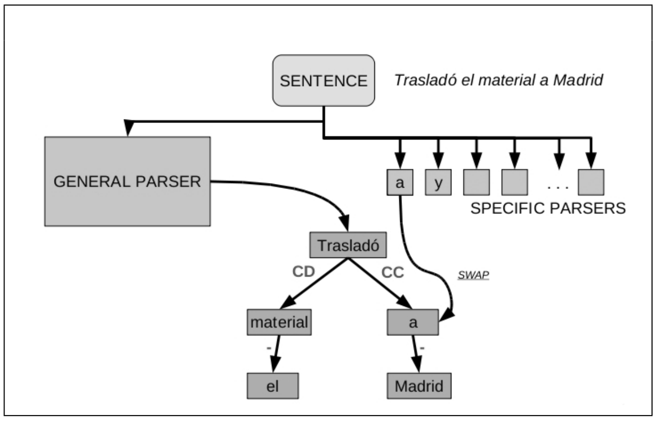

Figure 4: Swapping the node containing the preposition 'a' given by both parsers.

6. The algorithm returns the whole dependency tree that is produced from the suitable combination of the dependency trees given by the general and the specific parsers.

By applying the algorithm described we find that our n–version system parsed incorrectly 55 conjunctions while the general parser by itself parsed 56 conjunctions incorrectly. For the preposition "a" we get similar results: our n–version system parsed incorrectly 50 prepositions and the general one parsed 48 prepositions incorrectly.

These very first negative results do not mean that the n–version technique should be rejected. An explanation for this problem can be found in the AnCora corpus. This corpus was built automatically with a strong linguistic and manual validation step, but we have identified some errors that still remain. For instance, we realized that two sentences with the same syntactic structure can be found in AnCora with different annotations, as shown in Figure 5. The sentences of the example are the following: *La prensa mostró su afecto a los candidatos [The press showed its affection towards the candidates]* and *La telefonía permitió abrir el mercado a operadores*

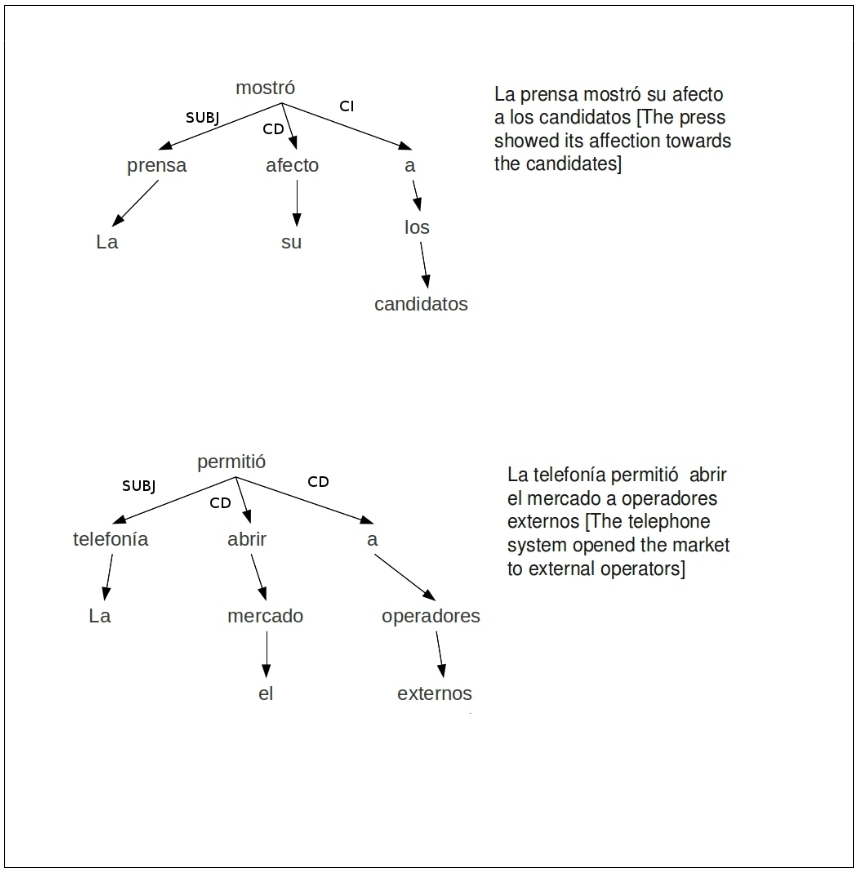
mostró
SUBJ
CD
CI
prensa
afecto
a
La
su
los
candidatos
La prensa mostró su afecto a los candidatos [The press showed its affection towards the candidates]
permitió
SUBJ
CD
CD
telefonía
abrir
a
La
mercado
operadores
el
externos
La telefonía permitió abrir el mercado a operadores externos [The telephone system opened the market to external operators]

Figure 5: Two sentences with the same syntactic structure but showing different annotations in AnCora: *La prensa mostró su afecto a los candidatos [The press showed its affection towards the candidates]* and *La telefonía permitió abrir el mercado a operadores externos [The telephone system opened the market to external operators]*

externos [The telephone system opened the market to external operators]. Since our approach sends both sentences to the same specific parser when detecting the preposition "a", one of them is parsed wrongly, according to the current version of AnCora, but actually it is objectively well parsed.

A second example of an incorrect annotation in AnCora is the one shown in Figure 6. The indirect object (CI) when the preposition "a" is involved is found in the following sentence: *Fox sólo podrá vencer a Labastida [Fox will only be able to beat Labastida].* This sentence has as its indirect object, the subsentence *a Labastida*, but in AnCora the node containing the preposition "a" is tagged as a direct object (CD). When our algorithm finds the preposition "a" in such a context then it considers any subtree that has "a" as its root to be an indirect object. Thus, once again we have a sentence that is formally well parsed by the n–version parser but marked as wrong in the testing process because the test corpus is not properly annotated.

At this point the importance of error free annotated corpora becomes obvious. Against the difficulty of having an error free annotated corpus, we demonstrated how simple is to build small and well–controlled sub-corpora for training specific parsers.

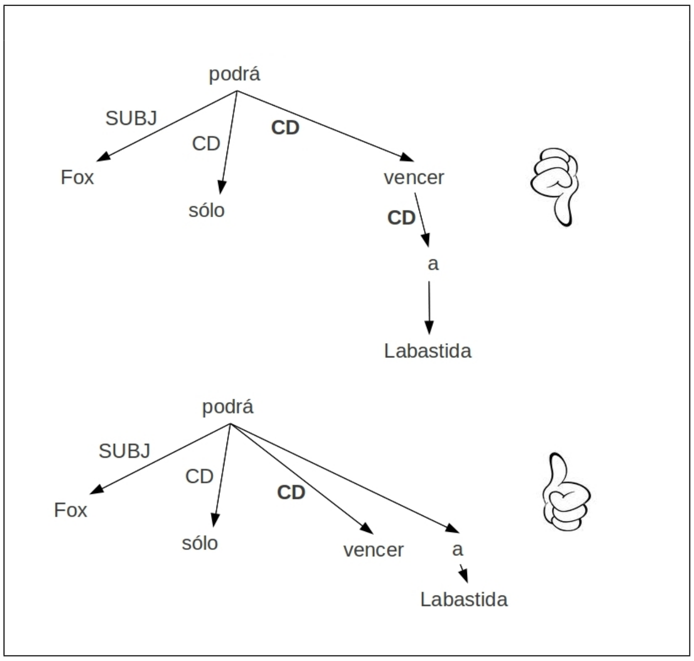

Figure 6: Bad and good ways of parsing the sentence *Fox sólo podrá vencer a Labastida [Fox will only be able to beat Labastida].*

4 Conclusions and Future Work

Since dependency parsing has become a demanded tool as a subsystem of more complex ones, a high accuracy is a very desirable feature for these parsers. But after several competitive evaluations of multilingual dependency parsers (CoNLL), accuracy seems to have reached a high but unbeatable limit (or at least, very difficult to beat). A number of reasons justify this behavior of dependency parsers, such as the quality of the tagging of training corpora or the tuning of parsers. Usually, these aspects are out of control, except if you can correct the annotation of a (very big) treebank or you can modify the source code or some other parameters of a parser. This is because we propose here our n–version approach, that permits to increase dependency parsing accuracy while avoiding unaddressable problems at a reasonable cost.

As said before, the experiments presented in this chapter are a first approach to the problem, so a more in–depth work will be done to get better results. This work includes not only the identification of more specific parsing cases but also the refinement of the programs that send each word to the most appropriate specific parser, making them more accurate. But we are conscious that the n–version parsing system that we propose here is not the perfect solution, and even when finished it will not reach 100% accuracy. It does not mean that it is lost work but a complement to other actions that could be accomplished, such as the improvement of the quality of the annotation of dependency treebanks or the refinement and tuning of parsing systems. Moreover, for future work we have planned to start using MaltOptimizer (Ballesteros & Nivre, 2012a; Ballesteros & Nivre, 2012b) in order to speed up the optimization processes.

Finally, we want to remind that even when we developed our work with MaltParser for Spanish, it could be carried out with other similar systems and/or languages.

Acknowledgments

This research is funded by the Spanish Ministry of Education and Science (TIN2009-14659-C03-01 Project), Universidad Complutense de Madrid and Banco Santander Central Hispano (GR58/08 Research Group Grant).

References

Ballesteros, M., Bautista, S., & Gervás, P. (2010a). Text Simplification Using Dependency Parsing for Spanish. In *KDIR–Knowledge Discovery and Information Retrieval 2010* (pp. 330–335). Valencia, Spain: INSTICC.

Ballesteros, M., Francisco, V., Díaz, A., Herrera, J., & Gervás, P. (2011). Inferring the Scope of Negation and Speculation Via Dependency Analysis. In *KDIR–Knowledge Discovery and Information Retrieval 2011* Paris, France: INSTICC.

Ballesteros, M., Herrera, J., Francisco, V., & Gervás, P. (2010b). A Feasibility Study on Low Level Techniques for Improving Parsing Accuracy for Spanish Using Maltparser. In S. Konstantopoulos, S. Perantonis, V. Karkaletsis, C. D. Spyropoulos, & G. Vouros (Eds.), *Artificial Intelligence: Theories, Models and Applications*, volume 6040 of *Lecture Notes in Artificial Intelligence* (pp. 39–48).: Springer.

Ballesteros, M., Herrera, J., Francisco, V., & Gervás, P. (2010c). Improving Parsing Accuracy for Spanish using Maltparser. *Journal of the Spanish Society for NLP (SEPLN)*, 44, 83–90.

Ballesteros, M. & Nivre, J. (2012a). MaltOptimizer: An Optimization Tool for MaltParser. In *Proceedings of the System Demonstration Session of the Thirteenth Conference of the European Chapter of the Association for Computational Linguistics (EACL 12)*.

Ballesteros, M. & Nivre, J. (2012b). MaltOptimizer: An Optimization Tool for MaltParser. In *In Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC 12)*.

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *COLING* (pp. 89–97).

Bosco, C., Montemagni, S., Mazzei, A., Lombardo, V., dell'Orletta, F., Lenci, A., Lesmo, L., Attardi, G., Simi, M., Lavelli, A., Hall, J., Nilsson, J., & Nivre, J. (2010). Comparing the influence of different treebank annotations on dependency parsing. In *LREC*.

Buchholz, S. & Marsi, E. (2006a). CoNLL–X shared task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL–X)* (pp. 149–164).

Buchholz, S. & Marsi, E. (2006b). Conll-x shared task on multilingual dependency parsing. In *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning* (pp. 149–164). Morristown, NJ, USA: Association for Computational Linguistics.

Chen, W., Kazama, J., Uchimoto, K., & Torisawa, K. (2009). Improving dependency parsing with subtrees from auto–parsed data. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2*, EMNLP '09 (pp. 570–579). Morristown, NJ, USA: Association for Computational Linguistics.

Cheng, Y., Asahara, M., & Matsumoto, Y. (2006). Multi-lingual dependency parsing at NAIST. In *CoNLL-X '06: Proceedings of the Tenth Conference on Computational Natural Language Learning* (pp. 191–195). Morristown, NJ, USA: Association for Computational Linguistics.

Civit, M., Martí, M. A., & Bufí, N. (2006). Cat3lb and cast3lb: From constituents to dependencies. In *FinTAL* (pp. 141–152).

Corston-Oliver, S. & Aue, A. (2006). Dependency parsing with reference to slovene, spanish and swedish. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06 (pp. 196–200). Stroudsburg, PA, USA: Association for Computational Linguistics.

Cortes, C. & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, 20(3), 273–297.

Councill, I. G., McDonald, R., & Velikovich, L. (2010). What's great and what's not: learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10 (pp. 51–59). Stroudsburg, PA, USA: Association for Computational Linguistics.

Cui, H., Sun, R., Li, K., yen Kan, M., & seng Chua, T. (2005). Question answering passage retrieval using dependency relations. In *In SIGIR 2005* (pp. 400–407).: ACM Press.

Culotta, A. & Sorensen, J. (2004). Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume* (pp. 423–429). Barcelona, Spain.

Ding, Y. & Palmer, M. (2005). Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05 (pp. 541–548). Morristown, NJ, USA: Association for Computational Linguistics.

Dreyer, M., Smith, D. A., & Smith, N. A. (2006). Vine parsing and minimum risk reranking for speed and precision. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06 (pp. 201–205). Stroudsburg, PA, USA: Association for Computational Linguistics.

Eisner, J. (1996). Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING–96)* (pp. 340–345). Copenhagen.

Herrera, J., Peñas, A., & Verdejo, F. (2005). Textual entailment recognition based on dependency analysis and *ordnet*. In *MLCW* (pp. 231–239).

Johansson, R. & Nugues, P. (2006). Investigating Multilingual Dependency Parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL–X)*.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., & Sontag, D. (2010). Dual decomposition for parsing with non-projective head automata. In *EMNLP* (pp. 1288–1298).

Lin, D. (1998). Dependency-based evaluation of MINIPAR. In *Proc. Workshop on the Evaluation of Parsing Systems*, Granada.

McDonald, R. & Nivre, J. (2007). Characterizing the errors of data–driven dependency parsing models. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* (pp. 122–131).: Association for Computational Linguistics.

McDonald, R. & Nivre, J. (2011). Analyzing and integrating dependency parsers. *Computational Linguistics*.

McDonald, R. T., Pereira, F., Ribarov, K., & Hajic, J. (2005). Non-projective dependency parsing using spanning tree algorithms. In *HLT/EMNLP*.

Nivre, J. (2006). *Inductive Dependency Parsing*. Text, Speech and Language Technology. Dordrecht: Springer.

Nivre, J., Hall, J., Kübler, S., McDonald, R., Nilsson, J., Riedel, S., & Yuret, D. (2007a). The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007* (pp. 915–932). Prague, Czech Republic: Association for Computational Linguistics.

Nivre, J., Hall, J., & Nilsson, J. (2004). Memory–based dependency parsing. In *Proceedings of CoNLL–2004* (pp. 49–56).: Boston, MA, USA.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kbler, S., Marinov, S., & Marsi, E. (2007b). Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2), 95–135.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., & Marinov, S. (2006). Labeled pseudo–projective dependency parsing with support vector machines. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL–X)* (pp. 221–225).

Palomar, M., Civit, M., Díaz, A., Moreno, L., Bisbal, E., Aranzabe, M., Ageno, A., Martí, M., & Navarro (2004). 3lb: Construcción de una base de datos de árboles sintáctico–semánticos para el catalán, euskera y español. In *Proceedings of the XX Conference of the Spanish Society for Natural Language Processing (SEPLN)* (pp. 81–88).: Sociedad Española para el Procesamiento del Lenguaje Natural.

Sagae, K. & Lavie, A. (2006). Parser combination by reparsing. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06 (pp. 129–132). Stroudsburg, PA, USA: Association for Computational Linguistics.

Shinyama, Y., Sekine, S., & Sudo, K. (2002). Automatic paraphrase acquisition from news articles. In *Proceedings of the second international conference on Human Language Technology Research* (pp. 313–318). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Snow, R., Jurafsky, D., & Ng., A. Y. (2005). Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Advances in Neural Information Processing Systems 17* Cambridge, MA.

Taulé, M., Martí, M., & Recasens, M. (2008). AnCora: Multilevel Annotated Corpora for Catalan and Spanish. In *Proceedings of 6th International Conference on Language Resources and Evaluation*.

Wu, Y., Lee, Y., & Yang, J. (2006). The Exploration of Deterministic and Efficient Dependency Parsing. In *Proceedings of the Conference on Computational Natural Language Learning (CoNLL–X)*.

Yamada, H. & Matsumoto, Y. (2003). Statistical dependency analysis with support vector machines. In *Proceedings of International Workshop of Parsing Technologies (IWPT'03)* (pp. 195–206).

Zeman, D. & Zabokrtský, Z. K. (2005). Improving Parsing Accuracy by Combining Diverse Dependency Parsers. In *Proceedings of the 9th International Workshop on Parsing Technologies*.