Dependency Parsing with Compression Rules

Pablo Gamallo
Centro Singular de Investigación en Tecnoloxías da Información - CITIUS
University of Santiago de Compostela, Galiza, Spain
pablo.gamallo@usc.es

Abstract

This article proposes a syntactic parsing strategy based on a dependency grammar containing both formal rules and a *compression* technique that reduces the complexity of those rules. Compression parsing is mainly driven by the single-head constraint of Dependency Grammar, and can be seen as an alternative method to the well-known *constructive* strategy. The compression algorithm simplifies the input sentence by progressively removing from it the *dependent* tokens as soon as binary syntactic dependencies are recognized. The performance of our system was compared to a deterministic parser based on supervised learning: MaltParser. Both systems were applied on several test sets of sentences in Spanish and Portuguese, from a variety of different domains and genres. Results showed that our parsing method keeps a similar performance through related languages and different domains, while MaltParser, as most supervised methods, turns out to be very dependent on the text domain used to train the system.

1 Introduction

For large scale applications in Information Extraction (IE), syntactic parsing should be robust, fast, and relatively accurate. Moreover, for specific IE applications such as semantic relation extraction, the output of parsing should be simple, easy to handle by the IE systems, and close to the semantic relationships to be extracted. For multilingual purposes, it is important to develop parsing techniques easily adapted to several languages. And finally, in order to be easily integrated in several NLP applications and tasks, the parsers should be applied on different text domains and genres with similar accuracy.

There are two well known approaches that could be considered as good approximations to the ideal system filling all these parsing properties: both deterministic dependency parsing (Nivre, 2004) and partial parsing using rule-based finite-state techniques (Abney, 1996). However, these two parsing approaches have still some problems.

Recent work on deterministic dependency parsing (called 'transition based') relies on supervised techniques requiring fully analyzed training corpora. Given that supervised techniques tend to have loss of precision when applied on texts of domains and genres different to those used for training (Rimell et al., 2009; Gildea, 2001), they need too much manual effort to create, adapt, or modify the training corpus to the target domain.

Speed is not actually a problem for finite-state techniques, which can parse large text corpora in a very efficient way. However, as they rely on complex rule-based notations, their main drawback is the difficulty to adapt such a rule system to different languages. Moreover, as most finite-state parsers are based on constituency grammars, their syntactic output cannot be easily integrated into IE applications. Unlike phrase constituents, dependencies are seen as simple and flat syntactic representations, very close to semantic relations which are the extraction target of many IE systems.

Many finite-state parsers are based on the *constructive* strategy (Grefenstette, 1996; Abney, 1996; Ait-Mokhtar and Chanod, 1997). In constructive parsing, an input sentence is manipulated by transducers that progressively transform the input with additional symbols encoding syntactic constituents or dependency relations (Oflazer, 2003). These transducers are pattern rules arranged in cascades: the output of a transducer is the input of the next one, which contains new rules adapted to the symbols added to the input.

So, parsing consists in transforming a basic input string into a more complex one by incrementally adding new symbols, bracket delimiters, labels, or special markers. This strategy incrementally constructs the linguistic representation within the input string, by making use of rules organized at different levels of complexity.

In this article, we propose a new (rule-based) finite-state parsing strategy based on dependencies, which minimizes the complexity of rules by using a technique we call *compression*. Compression parsing is driven by the ”single-head” constraint of Dependency Grammar, and can be seen as an alternative method to the *constructive* strategy. It simplifies the input string by progressively removing the *dependent* tokens as binary syntactic dependencies are recognized. At the end of the compression process, if all the dependencies in the sentence were recognized, the input string should contain just one token representing the main head (i.e., the *root*) of the sentence. This strategy was inspired by the *Right* and *Left* Reduce transitions used in deterministic dependency parsing. The input sentence is assumed to be tagged and disambiguated with a Part-Of-Speech (PoS) tagger. Moreover, the cost of manually creating rules can also be reduced by providing a suitable rule notation for linguists.

As in Ait-Mokhtar and Chanod (1997), we hold that the ordering of rules/transducers is in itself a genuine linguistic task, which must preserve the basic principle of doing the easiest task first (Abney, 1996). If rules (and so the grammar) are written following this principle, it is possible to use finite-state automata to deal with embedding structures and long-distance dependencies. Note that this is a deterministic parsing strategy, since it cannot produce ambiguous structures. The use of grammars in recent dependency parsers is almost non-existent. Our work propose to incorporate more linguistic knowledge into the parsing systems *via* light-weight grammars.

Finally, a system based on the compression strategy was implemented and released under General Public License. In addition, we defined a high level grammar language to define dependency-based rules and developed a grammar compiler to generate compression parsers in several languages (Gamallo and González, 2011). The performance of this system was compared to MaltParser (Nivre et al., 2007b), a deterministic parser based on supervised learning. Both systems were applied on several test sets of sentences of different domains and genres, in Spanish and Portuguese. One of the main motivations of the evaluation is to test whether the two systems are reliable to parse sentences of different domains and genres. It is generally accepted that supervised classifiers require some type of domain adaptation when both the training and test data sets belong to different domains. In particular, the accuracy of statistical parsers degrades when they are applied to different genres and domains (Rimell et al., 2009; Gildea, 2001). Results showed that our system keeps a similar performance through related languages and different domains, while MaltParser, as most supervised methods, is very dependent on the text domain used to train the system.

The remainder of this article is organized as follows. Section 2 introduces different approaches on both dependency and FST parsing. Then, Section 3 is focused on the description of our compression strategy. Next, Section 4 provides a general view of the implemented system. Then Section 5 reports the diverse experiments performed over the Portuguese and Spanish data sets. And finally, some conclusions are addressed in Section 6.

2 Related Work

Our strategy is based on both dependencies and FST parsing.

2.1 Dependency-based Syntactic Parsing

Following Nivre (2005), there are two traditions in dependency parsing: grammar-driven and data-driven parsing. Within each tradition, it is also possible to distinguish between two different approaches: non-deterministic and deterministic parsing. In the latest years, many works on dependency parsing have been developed within the approach to data-driven deterministic parsing, which is also known as *transition-based parsing*, in opposition to grammar-driven parsers. Other data-driven strategies are non-deterministic such as *graph-based dependency parsing* (McDonald and Pereira, 2006; Carreras, 2007; Martins et al., 2010) .

Transition parsing consists in inducing statistical models in combination with a deterministic strategy based on shift-reduce parsing (Nivre, 2004; Yamada and Matsumoto, 2003; Gómez-Rodríguez and Fernández-González, 2012). In

Nivre et al. (2004), the parsing strategy uses the arc-eager algorithm. In Gómez-Rodríguez et al. (2014), this algorithm is simplified by just using two transitions on undirected dependencies (the head-dependent information is erased), so as to avoid error propagation.

As we will show later, the main problem of the supervised learning strategies arises when the test sentences belong to linguistic domains very different from those found in the training corpus. We will show later the negative effect on system performance when the test and training data sets does not belong to the same domain and genre. As hand labeling data in new domains is a costly enterprise, the domain adaptation problem is a fundamental challenge in machine learning applications. Note that many NLP annotated resources are based on text from the news domain (in most cases, the Wall Street Journal), which is a poor match to other domains such as biomedical texts, electronic mails, transcription of meetings, administrative language, etc. (Daumé-III and Marcu, 2007; Daumé-III, 2006).

2.2 Finite-State Parsing Techniques

Finite-state technology has attractive properties for syntactic parsing, such as conceptual simplicity, flexibility, and efficiency in terms of space and time. It allows to build robust and deterministic parsers. Most finite-state based parsing strategies use cascades of transducers and are known as *constructive* parsers.

Parsing based on cascades of finite-state transducers can be viewed as a sort of string transformation. Finite-state transducers introduce progressively markings and labels within the input text. Transducers are arranged in cascades (or layers), where the subsequent transducer takes the output of the previous one as input. After a certain number of cascades, the initial input (which is a tagged sentence) is transformed into a structured text enriched with syntactic marks, such as chunk boundaries, labels for heads, special markers for functions or for relations between heads, etc. This strategy, known as *constructive*, progressively constructs the linguistic representation within the input string, by making use of rules/transducers organized at different levels (or layers) of complexity.

Most of finite state strategies aim to construct, not dependency graphs, but phrase based structures (Ait-Mokhtar et al., 2002; Ciravegna and Lavelli, 2002; Kokkinakis and Kokkinakis, 1999; Ait-Mokhtar and Chanod, 1997; Abney, 1996; Joshi, 1996; Grefenstette, 1996). In general, the construction of these structures is performed with three main cascades/layer of rules: chunking, head recognition, and attachment. The first layers of rules transform the tagged input into sequences of symbols representing basic chunks. Then, further rules take those chunks as input to add new symbols marking the heads of each chunk and, finally, new rules are applied on the output of the previous ones to annotate the identified heads with labels of syntactic functions (attachment).

The number of finite-state approaches focused on constructive dependency parsing is much smaller. We can merely cite the work by Oflazer (2003), where the input string is progressively enriched by additional symbols encoding dependency relations between words.

The finite-state strategy often relies on one fundamental property: *easy-first parsing*. Easy-first parsing means that the simplest tasks must be done first, leaving the harder decisions for the last steps of the parsing process. Parsing proceeds by growing *islands of certainty* (Abney, 1996; Eisner and Smith, 2010; Goldberg and Elhadad, 2010; Tratz and Hovy, 2011; Versley, 2014).

Finite-state parsers are the fastest systems among those achieving linear time complexity. So, they are scalable as the input text increases in size and are easily integrated into IE applications exploring the Web as corpus.

3 A Compression Parsing Strategy

We propose yet another FST based method, very similar to the constructive approaches, but by making use of a similar strategy to the shift-reduce algorithm as in incremental parsing. We call it *compression parsing*. It consists of a set of transducers/rules that *compress* the input sequence of tokens by progressively removing the dependent tokens as soon as dependencies are recognized. So, at each application of a rule, the systems reduce the input and make it easier to find new dependencies in further rule applications. In particular, short dependencies are recognized first and, as a consequence, the input is simplified so as to make lighter the recognition of long distance dependencies. This is inspired by the easy-first strategy.

The input of our parsing method is a sequence of disambiguated tagged tokens, where each token is associated with two pieces of information: a PoS tag representing the basic morpho-syntactic category of the token (NOUN, VERB, PRP, etc.) and a feature structure containing other relevant information of the token: morphological information (number, tense, person, etc.), lemma, token string, token position, etc. Tagged tokens are the elementary objects of the parsing process, while rules, which are implemented as finite state transducers, operates on tagged tokens. More precisely, rules successively identify dependencies between tokens, remove the dependents (if required) from the input sequence, and update (if required) the feature structures of the heads.

3.1 Description of rules

A rule is a tuple $< P, Arc, \triangle, Reduce >$, where:

- P is a pattern of tagged tokens, defined as a regular expression, whose general form is $\alpha X \beta Y \gamma$. X and Y are non-empty strings representing two tagged tokens, considered as the *core elements* of the pattern; while α, β, and γ represent the left, middle, and right contexts, respectively, of the core elements; they may be empty.

- Arc is the action that creates a dependency link between the core elements (X and Y), when a subsequence of the tagged input is matched by the pattern; two types of arcs are distinguished: $Left\_Arc$ adds a dependency link between X and Y, being X the dependent and Y the head; $Right\_Arc$ creates a dependency link between X and Y, being X the head and Y the dependent. This action also assigns a label (subject, modifier, adjunct, etc) to the dependency.

- $\triangle$ is a set of operations (*Agreement*, *Add*, *Correction*, *Inherit*, etc.) that are applied on the feature structure of the core elements; they can be used to perform very different actions: verifying if the two core elements (i.e., head-dependent) share a set of feature-values, adding new feature-values to the head, modifying some values of the head, correcting PoS tags, allowing the head to inherit selected values from the dependent, etc. *Add* and *Inherit* can be seen as constructive operations.

- $Reduce$ is the action that removes the *dependent* from the input string; this action can be suspended if the dependent token is likely to be the head in other dependencies that has not been recognized yet. So, the dependent will not be reduced until all its potential dependent tokens have been recognized.

Compressing rules, not only reduce the complexity of the search space (or input) of the remaining rules to be applied, but also construct relevant information (by adding linguistic features) for the application of those rules. In particular, it may store in the head relevant information of the removed dependent (*Inherit* operation), it permits to generate new attributes or modify values from existing attributes (*Add* operation), and also it can correct odd tagged PoS tags (Correction) (Garcia and Gamallo, 2010). In sum, the main contribution of our work is to define compressing rules as the integration of two parsing techniques: both transition-based and constructive parsing. On the one hand, the rules reduce the search space by removing the dependent tokens and, on the other hand, they can add relevant information to the head tokens by making use of operations such as *Add* or *Inherit*. Rules compresses the input sequence of tokens so as to make it easier the identification of more distant dependencies.

The *Inherit* operation is one of the most innovative contributions of our rule system. It allows transferring linguistic features to heads before removing the dependent tokens from the search space. This can be considered as one of the main contributions of our dependency-based strategy. In combination with *Add* operation, *Inherit* is useful to transfer relevant information from auxiliary, light, or modal verbs to their main verbs. It can also be used to model coordination by transfering categorial information from the coordinated structures to the coordinator, so as to make it possible subject-verb agreements. For instance, in the sentence ‘Paul and Mary are eating’, the *Inherit* operation allows the coordinator ‘and’ to inherit the nominal category of their parts and, by means of the *Add* operation, we can assign it the *plural* number. In addition, *Inherit* can also be used to transfer relevant morphological information (third person, plural, present tense) to the root verb ‘eat’ from the auxiliar ‘are’. This way, there is grammatical agreement between ‘(are) eating’ (3rd person and plural) and its subject ‘Paul and Mary’. As far as we know, no dependency grammar/parser has proposed this solution to the dependency-based analysis of verbal periphrases and coordinations.

3.2 Rule Ordering: Easy First

As was mentioned above, the ordering of rules in a FST parser is a genuine linguistic task. Rules are ordered in such a way that the easiest tasks, for instance short dependencies, are performed first. As in (Eisner and Smith, 2010; Goldberg and Elhadad, 2010; Tratz and Hovy, 2011; Versley, 2014), we assume that correct parsers exhibit a short-dependency preference: a word's dependents tend to be close to it in the string. The fact of identifying first easy syntactic structures, such as those ruled by modifiers and specifiers, allows us to easily find later distant links, for instance those relating verbs with subordinate conjunctions. Let us take the expression: 'if the former president of USA says ...'. There is here a long distance dependency between the verb 'says' and the conditional conjunction 'if'. In most sentences, both words are not adjacent since a great variety of tokens can be interposed. However, in a compression approach, we can guess that dependency by making use of a very simple pattern consisting in a subordinate conjunction (type:S) appearing immediately to the left of a verb (last rule T_6 below in 1). We just need the following sequence of transductions/rules[1]:

$$
\begin{aligned}
T_1 &: \quad \texttt{PRP} \leftarrow \texttt{PRP} \quad \texttt{NOUN} \\
T_2 &: \quad \texttt{NOUN} \leftarrow \texttt{ADJ} \quad \texttt{NOUN} \\
T_3 &: \quad \texttt{NOUN} \leftarrow \texttt{DT} \quad \texttt{NOUN} \\
T_4 &: \quad \texttt{NOUN} \leftarrow \texttt{NOUN} \quad \texttt{PRP} \\
T_5 &: \quad \texttt{VERB} \leftarrow \texttt{NOUN} \quad \texttt{VERB} \\
T_6 &: \quad \texttt{VERB} \leftarrow \texttt{CONJ<type:S>} \quad \texttt{VERB}
\end{aligned} \tag{1}
$$

In Figure 1, we show the application of the six rules on the input expression, as well as the effect of the *Reduce* transition at each level (for the sake of simplicity, label assignment is not taken into account).

Each rule processes the input from left to right repeatedly as long as new dependencies satisfying the pattern are found. Rules are checked top-down following the rank imposed by the linguist. When

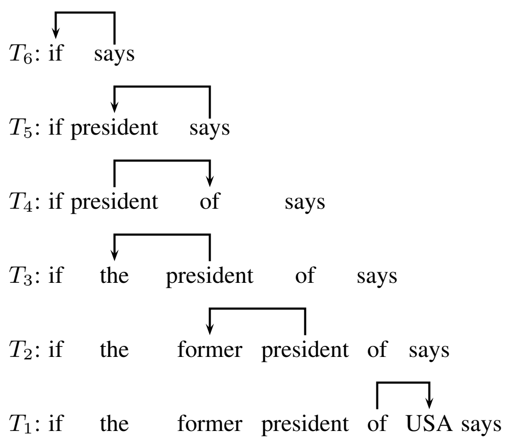
T_6: if says
T_5: if president says
T_4: if president of says
T_3: if the president of says
T_2: if the former president of says
T_1: if the former president of USA says

Figure 1: The six levels of analysis of 'if the former president of USA says...'

the parser reaches the last rule of the ranked list, if at least one dependency has been identified, the parser starts again from the beginning until no new dependency is found. So, the parser works iteratively until no change is made.

4 The Implementation

4.1 The Modules

Our compression parsing strategy has been inserted into a more generic natural language architecture, which consists of the following modules:

A set of PoS tagging *adapters* that modify the output of two PoS taggers, namely FreeLing (Padró and Stanilovsky, 2012) and Tree-Tagger (Schimd, 1995), so as to generate an unified PoS tagged format. The result of this process is the input of compression parsers. A set of grammars written with a specific grammar notation. The grammar formalism was described in (Gamallo and González, 2011). A grammar compiler, written in Ruby, that takes a particular grammar as input and generates a compression parser, written in Perl. An a set of multilingual parsers, generated by the compiler from various grammars.

The whole system, called DepPattern, is released under the GNU General Public License (GPL). [2]. Five parsers were generated for the following languages: Spanish, Portuguese, English, French, and Galician. The parsers are robust and very efficient: they are able to parse about 3,000

[1]To simplify, rule notation is focused on just the final *Reduce* operation

[2]`http://gramatica.usc.es/pln/tools/deppattern.html`

words per second on a processor Core 2 Quad, 2.8 Ggz. The system can be run on any GNU/Linux distribution. DepPattern was used for several web-based IE applications, namely Open Information Extraction from Wikipedia (Gamallo et al., 2012), extraction of semantic relations with distant supervision (Garcia and Gamallo, 2011), and extraction of bilingual terminologies from comparable corpora (Gamallo and Pichel, 2008). It has also been integrated into commercial tools, e.g. Linguakit[3] and Avalingua[4].

4.2 Multilingual Parsing

The parsers were developed for five languages: Spanish, Portuguese, Galician, French, and English. However, we have just written two small, and not very different, grammars:

Romance Grammar A grammar with 96 rules that are all applied to four Latin languages, namely, Spanish, French, Galician, and Portuguese.

English Grammar A grammar with 104 rules for English.

The cost of writing these two grammars is quite low. They are small and almost identical, since they share most rules except a reduced group of them which is specific for English. The strategy we followed to write grammars is based on two methodological principles: to start with rules with high coverage, and to start with rules shared by as many languages as possible

The objective is to find a trade-off between high performance and low effort, i.e. we look for efficiency. Most DepPattern rules of our grammars satisfy these two principles, giving rise to broad-coverage parsers. The quality of the French grammar is not good enough since this Latin language is quite different from Spanish, Portuguese, and Galician. There are important rules specific for French which have not been integrated in the shared grammar, for instance rules for dealing with partitives. Besides, we have not defined non-projective rules since, in general, they have low coverage and are language-dependent. Finally, the grammars also contain lexicalized rules for prepositions, modal verbs, quantifiers, etc. However, as the sets of lexical units are declared in external configuration files, the grammars are not required to be modified.

[3]`https://linguakit.com/`

[4]`http://cilenis.com/en/avalingua/`

5 Experiments

Our objective is to compare the performance of our FST-based strategy with that of a transition-based system. For this purpose, we compare DepPattern with MaltParser[5] (Nivre et al., 2007b), one of the top performing systems in the CoNLL shared tasks on multilingual dependency parsing in 2006 and 2007 (Hall and Nilsson, 2006; Nivre et al., 2007a). Experiments were performed on two languages: Portuguese and Spanish. The reason of making experiments on only two languages is the very high cost required to adapt the outputs of our system to the test data set.

5.1 Training and Test Corpora

The experiments were performed by making use of two dependency treebanks: Spanish Ancora 2.0 (Recasens and Martí, 2010) and Portuguese Bosque 8.0 (Afonso et al., 2002), also used by participants at the CoNLL 2006 shared task. In order to have a similar experimental setup for the two languages, each corpus was divided in a training part containing $115,000$ words and a test set with $100,000$ words. The two training corpora were only used to build the statiscal model of MaltParser. DepPattern, which is a grammar-based system, does not require any training corpus.

5.1.1 MaltParser's Optimization

We used MaltParser 1.7.1, equipped with nine different transition-based parsing algorithms. To select the best algorithm running in linear time (5 out 9: Nivre eager, Nivre standard, Stack proj, planar, and 2-planar), we validated the target algorithms on the Portuguese test data set. The best performance was achieved by 2planar (Gómez-Rodríguez and Nivre, 2010), based on arc-eager algorithm, even if the difference among the five tested algorithms was not statistically significant. So, the 2planar strategy was chosen to be compared with DepPattern.

All experiments were performed using a classifier based on support vector machines, as implemented in the LIBSVM package (Chang and Lin, 2001). Finally, we have reused the PoS tags and feature representation that produced the best results on Portuguese and Spanish for the MaltParser

[5]`htpp://www.maltparser.org/`

system at CoNLL shared task.

5.1.2 Test data sets

The two main data sets are the following:

bosque-test More than 3,000 sentences with 100M tokens taken from the Portuguese treebank.

ancora-test More than 3,000 sentences with 100M tokens taken from the Spanish treebank.

In order to compare MaltParser with DepPattern, we must address three problems when preparing the test data set:

- The tagsets and feature structures required by DepPattern are not the same as those available in the treebanks to train MaltParser.
- Each treebank was annotated following different criteria regarding some linguistic phenomena. Besides, there are also differences of criteria between DepPattern and some dependency solutions found in the treebanks.
- MaltParser can take advantage of the fact that both training and test sets belong to the same domain.

DepPattern requires as input PoS tagged text with the format provided by Tree-Tagger or FreeLing. Ancora 2.0 was annotated with FreeLing tags, but it is not the case of Bosque 8.0. In order to permit DepPattern to have the same entry as MaltParser for the Portuguese corpus, the PoS tags and features of Bosque 8.0 were converted into FreeLing-based PoS tags. Note that this PoS tag conversion allows us to apply MaltParser, in combination with FreeLing, to raw text. First, the PoS tags and feature representation of the two training sets were transformed into the FreeLing format. Then, the best MaltParser configuration was trained on these two sets and, finally, it was applied on text previously tagged with FreeLing. The results obtained were very similar (even slightly better) to those obtained with the PoS tags and features of the original treebanks. Conversion scripts are freely available.[6]

However, the main problem to evaluate DepPattern and to compare it with other systems is to minimize the noise derived from the choice of different linguistic criteria. In particular, we found in DepPattern grammars several linguistic decisions to define dependencies that are very different from those found in the exploited treebanks. For instance:

- All clause arguments (including the subject) are dependent on lexicalized verbs, and not on light or auxiliary verbs.
- All members of a coordination are dependent on the conjunction and not on the first coordinated member.

DepPattern grammars follow these two criteria. Bosque 8.0 follows neither of the two, and Ancora 2.0 just follows the first one. So, there are three different dependency criteria for dealing with these two linguistic phenomena, which have large coverage. In addition, there are many other conflicts to be considered: DepPattern analyzes the Portuguese possesive expressions in a different way as the treebank annotators: in ‘a sua mulher’ (*his wife*), ‘sua’ is dependent of ‘mulher’ and ‘a’ of ‘sua’. However, in the Portuguese treebank, ‘a’ and ‘sua’ both depend of ‘mulher’. There are also different decisions for the internal dependencies of periphrastic verbal expressions: e.g. ‘tiene que ir’ (*have to go*). Particle ‘que’ (*to*) can be either dependent of ‘tiene’ or of ‘ir’. All these differences are a serious drawback for comparing our grammar-based parser against a corpus-driven system. To solve the problem, we adapted our generic Romance Grammar to each treebank. As a result, we generated both a Portuguese parser adapted to Bosque and a Spanish parser adapted to Ancora. Another solution would have been to create conversor scripts to automatically modify the treebanks. However, as the algorithms required are not trivial, in particular for complex phenomena such as coordination, this strategy was not considered.

The third problem is related to the content similarity between the training and the test corpora, which could benefit the data-driven system. Apart from the two data sets extracted from the treebanks, we also built a small gold standard, called *open-test*, by manually analyzing sentences extracted from different sources. The description of this new test data set is the following:

open-test 42 Portuguese sentences with about 1K words taken from different sources with a

[6] `http://gramatica.usc.es/~gamallo/resources/depcorpus-test.tgz`

	Precision	Recall	F-score
ancora-test (es)	84.6	79.7	82
bosque-test (pt)	84.1	79.2	81.6
open-test (pt)	86.2	76.6	81.1

Table 1: Evaluation of DepPattern (unlabeled dependencies)

	Precision	Recall	F-score
ancora-test (es)	85	85	**85**
bosque-test (pt)	88.2	88.2	**88.2**
open-test (pt)	81.3	81.3	**81.3**

Table 2: Evaluation of MaltParser (unlabeled dependencies)

variety of genres and domains: Wikipedia (scientific domain and encyclopedic genre), a Portuguese novel by Machado Assis (literature genre), documents of the European Commission (economy domain).

Both *bosque-test* and *ancora-test* consist of sentences belonging to the same journalist genre as those found in the training corpus. By contrast, *open-test* consists of miscellaneous sentences whose content is distributed through different genres and domains. So it can be considered as an open-content test set. The dataset is freely available[7]

5.2 Evaluation

To evaluate the performance of the two systems, it is necessary to take into account that DepPattern produces partial parses. This system assigns the dependency relation '0' (or Root) to all unattached tokens. So, it is relevant to make use of precision and recall, instead of accuracy, to measure the performance of the DepPattern parsers. For MaltParser, precision and recall are the same: these values correspond to the unlabeled attachment score (UAS).

Furthermore, as the dependency labels of DepPattern are very different from those found in the two treebanks, the evaluation was restricted to unlabeled dependencies. Parser evaluation is conducted by comparing the dependencies guessed by the system with manually revised dependencies. Given these two data sets, precision and recall are defined as in (Lin, 1998). As '0' values, associated to unattached tokens, are considered as not found dependencies, they are relevant to measure recall in DepPattern. Punctuation marks are ignored in our evaluation.

Tables 1 and 2 show the results obtained with DepPattern and MaltParser, respectively, when applied on the different testing sets. The results obtained by MaltParser represent in fact the *UAS* of the system, since precision and recall are the same. MaltParser outperforms DepPattern on both *bosque-test* and *ancora-test*.

The scores we obtained using MaltParser follow the same tendency (even if they are not identical) of those obtained at the CoNLL 2006 shared task, where the system achieved 91% accuracy on Portuguese and 85 on Spanish (for unlabeled attachment scores). In our experiments, MaltParser obtained 88.2% and 85, respectively. The differences between the Portuguese scores at CoNLL and those obtained in our experiment could derive from small changes in the optimization procedure, and from the size of the training corpus. Notice that the performance of MaltParser is quite different across our three data sets: 88.2% (bosque) 85 (ancora), and 81.3 (open). By contrast, DepPattern achieves similar results in all the tests.

In the content-open test, we observe that whereas MaltParser gets down from 88.2 accuracy to 81.3, DepPattern keeps a similar score in the three datasets. It seems that the change of domain affects the performance of the data-driven system. By contrast, the grammar-based parser keeps a similar performance across domains and genres. It seems to be also most stable across different languages: it achieves similar results in Spanish and Portuguese because, on the one hand, these languages are very close and, on the other, DepPattern only use those grammar rules shared by the two languages. However, it is not easy to explain why MaltParser behaves in a very different way on two languages which are very similar in terms of grammar.

Finally, we also verified whether the two systems are complementary. This was made by measuring the statistical correlation between the results obtained by the two types of parsers. For this purpose, we analysed the Pearson correlation between the parses resulting from both DepPatter and MaltParser, in order to verify if they tend to make the same correct decisions. The Pearson coefficient obtained was low, namely 0.14, with only 69% of shared correct decisions. This

[7]`http://gramatica.usc.es/~gamallo/resources/depcorpus-test.tgz`

means that the two systems are complementary since many of the correct dependencies they guess are not the same. In sum, they are good in different ways. This insight essentially encourages to the pursuit of hybrid approaches and parser combinations, since both strategies seem to be complementary and should work together to produce more efficient results.

6 Conclusions

The work described in the article can be seen as a contribution to improve old parsing strategies introduced at the end of the twentieth century, when most efficient techniques were based on rules/FST and constructive approaches. In particular, we described a grammar-driven parser based on FST, called compression parsing, which takes into account some elements from deterministic and incremental dependency parsing, namely *Arc* and *Reduce* transitions. This compression method, implemented in DepPattern, was compared with a data-driven, transition-based system: MaltParser.

The cost and effort of developing compression parsers for several languages is not very high, since they can achieve reasonable performance using just very simple, multilingual, and general-purpose grammars. In this article, we have also introduced a simple methodology to write multilingual and general-purpose grammars.

In future work, it would be interesting to compare a variety of grammar-driven systems by measuring, not only their performance, but also the complexity of the underlying grammar: number of rules, size (in bytes) of the source files, etc. It should also be important to quantify and compare the cost and effort of both writing grammars and building treebanks. Moreover, to complement quantitative evaluation, it will be necessary to define objective protocols to compare parsers on the basis on qualitative evaluation (Lloberes et al., 2014).

Finally, we claim that FST-based parsing techniques simulate how we solve problems quickly, by taking first easy decisions which, in turn, make it easier to solve further complex tasks. However, these parsing techniques are far from simulating two other interesting cognitive operations: first, how grammars are learnt and, second, how sentences are processed. Data-driven approaches can be seen as a good approximation to the way humans learn grammars, while incremental left-to-right parsing can be seen as a simulation of how humans process and understand input sentences. Here, a question arises: would it be possible to define a method taking advantage of all those parsing strategies? In other words, could be it possible to model a strategy that learns grammar rules from data, orders them as cascades of progressively more complex transducers, and applies them to sentences in an incremental way? A method provided with these three ‘human properties’ would be closer to the canonical systems in Artificial Intelligence, since the main objective would be, not only to produce parse trees, but also to simulate how humans understand sentences.

Acknowledgement

This work has been supported by RedPLIR, Xunta de Galicia, ref: CN2014/034.

References

Steven Abney. 1996. Partial parsing via finite-state cascades. In *Proceedings of the ESSLLI’96 Robust Parsing Workshop*, pages 8–15.

Susana Afonso, Eckhard Bick, Renato Haber, and Diana Santos. 2002. Floresta sintá(c)tica”: a treebank for portuguese. In *LREC 2002, the Third International Conference on Language Resources and Evaluation*, pages 1698–1703, Las Palmas de Gran Canaria, Spain.

Salah Ait-Mokhtar and Jean-Pierre Chanod. 1997. Incremental finite-state parsing. In *Proceedings of the 5th Conference on Applied Natural Language Processing (ANLP-97)*, Washington, DC, USA.

Salah Ait-Mokhtar, Jean-Pierre Chanod, and Claude Roux. 2002. Robustness beyond shallowness: Incremental deep parsing. *Natural Language Engineering*, 8(2/3):121–144.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *EMNLP/CoNLL.*

Chih Chang and Chih J. Lin, 2001. *LIBSVM: a library for support vector machines.*

Fabio Ciravegna and Alberto Lavelli. 2002. Full parsing approximation for information extraction via finite-state cascades. *Natural Language Engineering*, 8(2/3):145–165.

Hal Daumé-III and Daniel Marcu. 2007. Frustratingly easy domain adaptation. In *45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Hal Daumé-III. 2006. Domain adaptation for statistical classifiers. *Artificial Intelliegence Research*, 26:101–126.

Jason Eisner and Noah A. Smith. 2010. Favor short dependencies: Parsing with soft and hard constraints on dependency length. In Harry Bunt, Paola Merlo, and Joakim Nivre, editors, *Trends in Parsing Technology: Dependency Parsing, Domain Adaptation, and Deep Parsing*, chapter 8, pages 121–150. Springer.

Pablo Gamallo and Isaac González. 2011. A grammatical formalism based on patterns of part-of-speech tags. *International Journal of Corpus Linguistics*, 16(1):45–71.

Pablo Gamallo and José Ramom Pichel. 2008. Learning Spanish-Galician Translation Equivalents Using a Comparable Corpus and a Bilingual Dictionary. *LNCS*, 4919:413–423.

Pablo Gamallo, Marcos Garcia, and Santiago Fernández-Lanza. 2012. Dependency-based open information extraction. In *ROBUS-UNSUP 2012: Joint Workshop on Unsupervised and Semi-Supervised Learning in NLP at the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, Avignon, France.

Marcos Garcia and Pablo Gamallo. 2010. Using morphosyntactic post-processing to improve postagging accuracy. In *PROPOR-2010*, Porto-Alegre, Brasil.

Marcos Garcia and Pablo Gamallo. 2011. Dependency-based text compression for semantic relation extraction. In *Workshop on Information Extraction and Knowledge Acquisition (IEKA 2011) at 8th International Conference on Recent Advances in Natural Language Processing (RANLP 2011)*, pages 21–28.

Daniel Gildea. 2001. Corpus variation and parser performance. In *2001 EMNLP Conference*, Pittsburgh, PA.

Yoav Goldberg and Michael Elhadad. 2010. An efficient algorithm for easy-first non-directional dependency parsing. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 742–750, Stroudsburg, PA, USA.

Carlos Gómez-Rodríguez and Daniel Fernández-González. 2012. Dependency parsing with undirected graphs. In *13th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 66–76, Avignon, France.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1492–1501, Stroudsburg, PA, USA.

Carlos Gómez-Rodríguez, D. Fernández-González, and V. Bilbao. 2014. Undirected dependency parsing. *Computational Intelligence*.

Gregory Grefenstette. 1996. Light parsing as finite-state filtering. In *Workshop on Extended Finite State Models of Language ECAI'96*, Budapest, Hungary.

Johan Hall and Jens Nilsson. 2006. CoNLL-X shared task on multilingual dependency parsing. In *The tenth CoNLL*.

Aravind Joshi. 1996. A parser from antiquity: An early application of finite state transducers to natural language parsing. In *ECAI-96 Workshop on Extendend Finite State Models of Languages*, Budapest, Hungary.

Dimitrios Kokkinakis and Sofie Johansson Kokkinakis. 1999. A cascaded finite-state parser for syntactic analysis of swedish. In *The 9th EACL*, Bergen, Norway.

Dekang Lin. 1998. Dependency-Based Evaluation of MINIPAR. In *Workshop on Evaluation of Parsing Systems*, Granada, Spain.

Marina Lloberes, Irene Castellón, Lluís Padró, and Edgar González. 2014. Partes. test suite for parsing evaluation. *Procesamiento del Lenguaje Natural*, 53:87–94.

André F. T. Martins, Noah A. Smith, Eric P. Xing, Pedro M. Q. Aguiar, and Mário A. T. Figueiredo. 2010. Turboparsers: Dependency parsing by approximate variational inference. In *Empirical Methods in Natural Language Processing (EMNLP'10)*, Boston, USA.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL-2006*, pages 81–88.

Joaquim Nivre, Johan Hall, and Jens Nilson. 2004. Memory-based dependency parsing. In *Proceedings of CoNLL*, pages 49–56.

Joakim Nivre, Johan Hall, Sandra Kübler, Ryan McDonald, Jens Nilson, Sebastian Riedel, and Deniz Yuret. 2007a. The CoNLL-2007 shared task on dependency parsing. In *Proceedings of the Shared Task Session of EMNLP-CoNLL 2007*, pages 915–932, Prague, Czech Republic.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007b. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):115–135.

Joaquim Nivre. 2004. Incrementality in deterministic dependency parsing. In *ACL Workshop on Incremental Parsing: Bringing Engineering and Cognition Together*, pages 50–57.

Joaquim Nivre. 2005. Dependency grammar and dependency parsing. Technical Report MSI report 05133, Växjö University: School of Mathematics and Systems Engineering.

Kemal Oflazer. 2003. Dependency parsing with an extended finite-state approach. *Computational Linguistics*, 29(4):515–544.

Lluís. Padró and Evgeny Stanilovsky. 2012. Freeling 3.0: Towards wider multilinguality. In *Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey.

Marta Recasens and M. Antònia Martí. 2010. AnCora-CO: Coreferentially annotated corpora for spanish and catalan. *Language Resources and Evaluation*, 315-345(4):315–345.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Conference on Empirical Methods in Natural Language Processing*, pages 813–821, Singapore.

Helmut Schimd. 1995. Improvements in part-of-speech tagging with an application to german. In *ACL SIGDAT Workshop*, Dublin, Ireland.

Stephen Tratz and Eduard Hovy. 2011. A fast, effective, non-projective, semantically-enriched parser. In *In Proceedings of EMNLP. Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio*.

Yannick Versley. 2014. Experiments with easy-first nonprojective constituent parsing. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 39–53, Dublin, Ireland.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistically dependency analysis with support vector machines. In *Proceedings of 8th International Workshop on Parsing Technologies (IWPT)*, pages 195–206.