Extracting LTAG Grammars from a Spanish Treebank

Prasanth Kolachina  
LTRC, IIIT-Hyderabad,  
Hyderabad, India  
prasanth_k@research.iiit.ac.in

Srinivas Bangalore  
AT&T Labs - Research,  
180 Park Ave, Florham Park  
NJ, USA  
srini@att.research.com

Sudheer Kolachina  
LTRC, IIIT-Hyderabad,  
Hyderabad, India  
sudheer.kpg08@research.iiit.ac.in

Abstract

Treebank grammars have been known to help in building robust, wide-coverage statistical parsers that also obtain state-of-art accuracies. In this work, we present a system that extracts LTAG grammars for Spanish from a constituency-based Spanish treebank. We evaluate the extracted grammar in terms of its size, its coverage on unseen data and the performance of a supertagger trained on it. The supertagger built using the MaxEnt framework achieves an error rate of 20.36% for a tagset containing 10,424 supertags.

1 Introduction

There have been a number of hand-crafted grammar development projects starting in the 1990s which have attempted to build comprehensive wide-coverage grammars for many natural languages using different grammatical formalisms such as LFG, TAG, CCG and HPSG. While these efforts contributed to computational grammars with detailed linguistic analysis, they have lacked the distributional information necessary for disambiguation tasks such as parsing. On the other hand, treebanks have annotated large amounts of text data in different languages with phrase structure and dependency information (Marcus et al., 1994; Hajič and Hladká, 1998; Begum et al., 2008). While these treebanks have been vital for the creation of robust parsers, grammars extracted from these treebanks for building parsers typically are not linguistically interesting. Therefore, there have been recent efforts to extract grammars in different grammatical formalisms from treebanks to recover linguistically meaningful units of the grammar. (Charniak, 1996) for context-free grammars, (Bangalore, 1997; Neumann, 1998) for lexicalized tree grammars, (Xia, 2001; Chen et al., 2006) for tree-adjoining grammars and (Hockenmaier, 2006; Hockenmaier and Steedman, 2007) for combinatory categorial grammars and (Miyao et al., 2005) for HPSG grammars are only a few examples of them.

Automatically extracted grammars have been effectively used in a number of applications. The MICA parser uses a phrase-structure tree insertion grammar extracted from the Penn treebank (Bangalore et al., 2009) to derive the dependency structure of a sentence. Automatically extracted grammars have been used to train statistical parsers to improve the coverage of the lexicon. Lexicalized grammars extracted from the treebank have been used to train supertaggers (Bangalore and Joshi, 1999; Clark, 2002) which in turn have been shown to be useful in a wide variety of NLP applications. Supertaggers assign a sequence of tags for a given sentence, similar to pos-tagging but possessing relatively richer syntactic information in comparison to part-of-speech taggers. Automatically extracted grammars have also been used to evaluate the coverage of hand-crafted grammars (Xia and Palmer, 1999) and to detect certain types of errors in treebank annotation (Xia et al., 2000b).

In this paper, we extract an LTAG grammar from a constituency-based treebank for Spanish and create a supertagger lexicon. We evaluate the resulting grammar in terms of size, its coverage on unseen data and the performance of a supertagger. The treebank used for the grammar extraction is currently the largest available resource for the Spanish language. The extracted supertags can be used in context-based models in machine translation and for wide-coverage statistical parsing.
The outline of the paper is as follows. We discuss a small volume of the prior work on grammar extraction closely related to the current work in Section 2. A brief outline of the LTAG formalism is provided in Section 3. In Section 4, the Spanish treebank and the annotation procedure used to annotate the corpus are discussed briefly. We present our algorithm for extracting LTAG grammars from the Spanish treebank in Section 5. We also discuss a few interesting issues in the development of the grammar extraction module. We evaluate the grammar extracted using our system and report the results of our evaluation in Section 6. Finally, we summarize our conclusions in Section 7.

2 Related Work

The LexTract system (Xia, 1999; Xia et al., 2000b; Xia, 2001) is a grammar development tool to extract lexicalized TAG grammars from bracketed corpora. The extraction procedure is carried out in three steps: in the first step, the distinction between argument, adjunct and modifier is introduced into the parse tree, followed by a factorization step in which the elementary trees are extracted. The final step involves filtering out the invalid elementary trees extracted from the corpora. The extracted structures in the grammar can be classified into one of the three categories: a) spine-etree b) mod-etree c) conj-etree. The extracted grammars are evaluated by finding out how many structures in the extracted grammar match the structures defined in a hand-crafted grammar. (Xia and Palmer, 2000) propose two comparison metrics, t-match and c-match to evaluate the extracted grammars. The t-match metric compare the tree structures from the two grammars whereas the c-match metric compares the components (sub-trees) in two structures. (Xia et al., 2000a) extract LTAG grammars from English, Chinese and Korean treebanks and compare the extracted grammars using the comparison metrics between each pair of languages.

In a closely related work, (Chen et al., 2006) use a bottom-up extraction algorithm as opposed to the top-down algorithm used in LexTract to extract elementary trees from the PENN treebank (PTB) corpus. The approach followed in this work has three primary steps: head percolation, distinction between argument and adjunct nodes followed by factorization into elementary structures. In the head percolation step, a head word is assigned to each node in the parse tree using local structural information. The head information derived thus, is used to determine the trunk of a tree defined as the path containing nodes labelled with the same head word. In the following step, the status of all the nodes in the parse tree is determined as either a complement or an adjunct. (Chen et al., 2006) present two different procedures to determine the status of a node. The first procedure, inspired by (Collins, 1997) uses the label and the semantic tags of a node and its parent to determine the node status. The other procedure used in (Xia, 1999) uses the label and the semantic tags of the node and its head sibling node along with the distance between both nodes to determine the node status. The final step in the extraction procedure involves factorizing the parse tree into elementary structures. A recursive procedure is used to extract trees bottom up given a particular treebank bracketing. The procedure extracts initial and auxiliary tree structures from the parse tree. The grammars extracted using this procedure are evaluated according to size, coverage on the test dataset and their performance in supertagging.

There has also been work on extracting grammars for formalisms other than LTAG. (Hockenmaier, 2006; Hockenmaier and Steedman, 2007) extract Combinatory Categorial grammars (CCGs) from treebanks for German and English. The procedure uses a translation algorithm to define a mapping between phrase structure trees and CCG derivations. The nodes in the parse tree are initially marked as arguments or adjuncts or head nodes. Once the classification is done, the parse tree is binarized by inserting dummy nodes into the tree such that all children to the left of the head node are found in the right sub-tree. The nodes in the binary tree are then assigned one of the following roles: a) root node b) head and complement c) head and adjunct. The assignment procedure is similar to a top-down normal-form derivation. The extracted grammar in (Hockenmaier, 2006) for German is evaluated by computing the number of sentences for which the extraction procedure fails and using metrics like lexicon size and coverage.

3 Lexicalized Tree Adjoining Grammars

Tree Adjoining Grammars (TAGs) were first introduced in (Joshi et al., 1975). The grammar is defined by specifying a finite set of elementary
trees and a set of composition operations. The set of elementary trees is further divided into two types: a) initial trees b) auxiliary trees (see Figure 1). TAG uses two composition operations to build complex structures: i) substitution that takes place only on non-terminal nodes of frontier of a tree ii) adjunction to build a new tree from an auxiliary tree and a tree in which the root and footer nodes of the auxiliary tree share the same label as the node of adjunction in the tree (shown in Figure 2). The structures built using the composition operations on the elementary structures are defined as derived trees and the dependencies as derivation trees.

In the rest of this section, we define the key properties of the LTAG formalism which is based on the earlier TAG formalism: lexicalization, Extended Domain of Locality (EDL) and factoring of recursion from the domain of dependencies. A detailed discussion about these properties is presented in (Schabes et al., 1988).

- **Lexicalization**: A grammar is said to be lexicalized if it consists of
  i) a finite set of elementary structures (strings, trees, directed acyclic graphs, etc), each structure anchored on a lexical item.
  ii) a finite set of lexical items, each associated with at least one of the elementary structures of the grammar and
  iii) a finite set of operations combining these structures.

This property establishes a direct link between the lexicon and the syntactic structures defined in the grammar.

- **Extended Domain of Locality (EDL)**: The Extended Domain of Locality (EDL) property has two parts:
  i) Every elementary structure must contain all and only the arguments of the anchor in the same structure and
  ii) For every lexical item, the grammar must contain an elementary structure for each syntactic environment the lexical item might appear in.

The part (ii) of the EDL property distinguishes LTAG from other grammar formalisms. This property ensures that there is one elementary tree for every syntactic environment that the anchor may appear in.

- **Factoring of Recursion**: Recursion is factored away from the domain for the statement of dependencies. Recursive constructs in TAG are represented as auxiliary trees, combining with elementary trees by the operation of adjunction. As a consequence of the fact that there is a ‘minimal’ elementary structure for every syntactic environment corresponding to each lexical item, the factoring of recursion property follows as a corollary of EDL property.

In our work, our goal is to extract grammars in the LTAG formalism for the Spanish language from a constituency annotated treebank.

4 **AnCora Spanish Treebank**

The AnCora project\(^1\) originally started as an effort aimed at the creation of annotated corpora for Catalan and Spanish (Civit and Martí, 2004). The AnCora corpus used in the annotation project consists of corpora for Catalan and Spanish, created by extending the already existing 3LB treebanks (Civit and Martí, 2004). The Spanish corpus was extended using texts from the EFE news

\(^1\)Official page: [http://clic.ub.edu/corpus/en](http://clic.ub.edu/corpus/en)
corpus and articles from the ‘El Periodico’ newspaper. Similarly, the annotation scheme used in the earlier work was enriched with semantic information. In this section, we only briefly mention the annotation details relevant for our discussion on extraction of LTAG grammars from the Spanish treebank. See (Martí et al., 2007; Taulé et al., 2008) for a detailed step-by-step account of the development of the treebanks.

The Spanish treebank contains annotations at different levels of linguistic analyses done using automatic, semi-automatic and manual methods of annotation depending on the kind of linguistic information being annotated. The following are the details about the different levels of annotation carried out on the raw corpus.

1. **Morphosyntactic annotation**: The corpus was annotated with POS tags, lemma and chunk information in this phase. The morphological annotation was done automatically using MACO (Atserias et al., 1998), an annotation tool for morphological analysis. The analyzer assigns all possible analysis to a word, which is subsequently disambiguated using a constraint-based probabilistic tagger RELAX (Padró, 1998). The disambiguated output is then passed to a automatic chunker to obtain a flat syntactic analysis, used in syntactic annotation later.

2. **Syntactic annotation**: The full syntactic annotation phase was carried out in two steps. After validating the output of the morphosyntactic annotation phase, all the syntactic constituents in a sentence were labelled. Since Spanish is a pro-drop language, cases of elliptical subjects were explicitly annotated in the treebank. The annotation scheme does not make any distinction between arguments and adjuncts. In the next step, the constituents were marked with syntactic functions.

3. **Semantic annotation**: Semantic annotation of the corpora included labelling the argument structure of verbal predicates where the relationship between arguments and the predicate was specified using PropBank-style annotation, named entity annotation and annotation of senses of all nominal phrases using WordNet. The annotation of verbal predicates was done semi-automatically.

We use the latest version of the Spanish treebank to extract the LTAG grammar for Spanish. Table 1 contains a summary of statistics from the Spanish treebank. The treebank contains 528,024 tokens (17,376 sentences), which is approximately about half the size of the Penn treebank. However, this is also currently the largest treebank available for the Spanish language. Figure 3 shows a sample parse tree found in the Spanish treebank. Also, it is necessary to mention that the version of the treebank used in our work does not contain the thematic role information for the verbal nodes in all the sentences.

(Civit et al., 2006) present a method to convert the constituency parse trees into dependency representa-

<table>
<thead>
<tr>
<th>Vocabulary</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td># of tokens in the treebank</td>
<td>528,024</td>
</tr>
<tr>
<td># of types in the treebank</td>
<td>46,493</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Annotation scheme</th>
</tr>
</thead>
<tbody>
<tr>
<td># of POS tags(coarse-grained)</td>
</tr>
<tr>
<td># of POS tags(fine-grained)</td>
</tr>
<tr>
<td># of syntactic tags</td>
</tr>
<tr>
<td># of function tags</td>
</tr>
<tr>
<td># elliptic subject nodes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Corpora</th>
</tr>
</thead>
<tbody>
<tr>
<td># of sentences</td>
</tr>
<tr>
<td>Average sentence length</td>
</tr>
</tbody>
</table>

Version 1.0.1. The treebank can be downloaded from the official project page.
resentations using manually created head tables and function tables\(^3\). The head table contains information to indicate which of the daughter nodes of a constituent is its head. The function table on the other hand, provides the label to be assigned to the edge between the head and the modifier nodes.

5 Grammar Extraction

In our current work, we modify the extraction procedure of Chen described in (Chen et al., 2006) to extract an LTAG grammar from the AnCora Spanish treebank. The entire extraction procedure for the sample parse tree in Figure 3 is shown in Figure 4.

Given a parse tree \(\gamma\) for a sentence S from the treebank, we refine the given parse tree by replacing the coarse-grained non-terminal labels with labels of increased granularity based on the morphological properties of nominal elements and the syntactic role of a clausal node in the sentence. A sample refined tree is shown in Figure 4. The spec node shown in Figure 3 is replaced by spec.fs using the gender and the number information. By doing this, we obtain a parse tree containing fine-grained labels making it easier to write rules for head computation. The refined treebank contains 83 node labels against the 24 labels assigned in the actual annotation process.

The next step is to percolate the head word of each node in the tree using local structural information. We use a manually constructed head table to compute the head words for each node. Each entry corresponding to a non-terminal in the treebank is associated with a set of rules in the decreasing order of precedence. The fine-grained labels from the treebank refinement step help in disambiguating nodes occurring with the same label in different contexts allowing the rules for head computation to be specified in a simple manner while taking the local structure information into account. The head rules written on top of this refined treebank contain an average of 8 rules to compute the head for each of the 83 node labels. Trunks corresponding to each word in the parse tree are determined by joining the nodes labeled with the same head word in a bottom-up fashion.

After the trunks corresponding to each of the lexical items in a sentence have been identified, it is necessary to determine the status of a node as an argument or an adjunct with respect to its parent node. An argument node that is immediately dominated by a node on the trunk would be placed on the frontier of the elementary tree associated with the parent node’s head word (say \(\alpha\)). Similarly, an adjunct node would belong to an auxiliary tree \((\beta)\) modifying the elementary tree \(\alpha\). If an adjunct node is mistakenly placed on the frontier of an elementary tree, it is a violation of the Extended Domain of Locality property of the TAG formalism. Arguments of verbal predicates in the Spanish treebank are annotated with thematic roles as explained in Section 4. In the case of sentential arguments, adjuncts were explicitly annotated to be so. On the other hand, predicates which sentential constructions as a part of their argument structure are assigned thematic roles, instead of adjunct annotation.

---

\(^3\)The converted dependency treebank is also available on the project page.
We use a four step procedure to determine the status of a node as an argument or adjunct. We use i) the thematic roles annotated in the treebank to determine the status of arguments of a verbal predicate ii) explicit adjunct annotation to determine the status of sentential arguments iii) function tags assigned to the constituents in case the thematic roles are missing iv) a manually constructed function table containing the dependency labels to determine the status of arguments of non-verbal elements. The function tags were required in the case where verbal arguments were not annotated with thematic roles. The earlier works on grammar extraction carried out on the Penn treebank do not use semantic information about the arguments in a parse tree to determine the argument/adjunct status of a node. We make use of this additional annotation available in the Spanish treebank to aid us in achieving elementary structures linguistically closer to those of a hand-crafted grammar. The head percolation and the argument/adjunct distinction steps corresponding to the example in Figure 3 are illustrated in Figure 4(b).

Algorithm 1 Factorization procedure.

Input: parse tree \( p \) and list of trunks \( T \) corresponding to each lexical item
Output: set of elementary trees \( I \)
1: \( \text{elementaryTrees} = \{\} \)
2: for all \( t \in T \) do
3: \( \text{node} = t[0] \)
4: if \( \text{node}.status == \text{complement} \) then
5: \( \text{etree} = \text{extractInitialTree}(p, t) \)
6: else if \( \text{node}.status == \text{adjunct} \) then
7: \( \text{etree} = \text{extractAuxiliaryTree}(p, t) \)
8: end if
9: \( \text{elementaryTrees}.add(\text{etree}) \)
10: end for
11: return \( \text{elementaryTrees} \)

The final step in the extraction procedure involves a bottom-up procedure to factorize a parse tree into a set of elementary trees. Algorithm 1 lists the outline of the factorization procedure. Given a parse tree \( p \), the trunk corresponding to each of the lexical items in the sentence are identified using the head word information from the Head Percolation step. The factorization procedure accepts the parse tree \( p \) and the list of trunks \( T \) as input. \( \text{elementaryTrees} \) maintains a list of all the TAG structures extracted from the parse tree. The extraction algorithm varies depending on the status of the head node on the trunk (lines 5 and 7). Argument nodes result in initial trees while adjunct nodes are factored separately into auxiliary trees.

Algorithm 2 Extract Initial trees procedure.

Input: parse tree \( p \) and trunk \( t \)
Output: elementary tree \( E \)
1: \( E = \text{initTAGTree}() \)
2: \( \text{depth} = 0 \)
3: for all \( \text{node} \in t \) do
4: \( E.\text{addNode}(\text{node}.\text{depth}) \)
5: \( \text{childNodes} = \text{node}.\text{getChildren}() \)
6: for all \( c \in \text{childNodes} \) do
7: if \( c \in t \) then
8: \( E.\text{addNode}(c, \text{node}) \)
9: \( E.\text{setHead}(c) \)
10: else if \( c.\text{status} == \text{complement} \) then
11: \( c.\text{setNodeType}(\text{substitution}) \)
12: \( E.\text{addNode}(c, \text{node}) \)
13: end if
14: end for
15: \( \text{depth} = \text{depth}+1 \)
16: end for
17: return \( E \)

Algorithm 3 Extract Auxiliary trees procedure.

Input: parse tree \( p \) and trunk \( t \)
Output: elementary tree \( E \)
1: \( E = \text{initTAGTree}() \)
2: \( \text{depth} = 0 \)
3: \( \text{node} = t[0] \)
4: \( d = \text{node}.\text{distanceFromHead}() \)
5: if \( d \geq 3 \) or \( \text{node}.\text{isTerminalNode}() \) then
6: \( E.\text{addNode}(\text{node}.\text{parent}().\text{depth}) \)
7: \( E.\text{setHead}(\text{node}) \)
8: \( \text{tempNode} = \text{node}.\text{parent}() \)
9: \( \text{tempNode}.\text{setNodeType}(\text{footer}) \)
10: \( E.\text{addNode}(\text{tempNode}.\text{depth}+1) \)
11: else
12: \( E.\text{addNode}(\text{node}.\text{head}().\text{depth}) \)
13: \( E.\text{setHead}(\text{node}) \)
14: \( \text{tempNode} = \text{node}.\text{head}() \)
15: \( \text{tempNode}.\text{setNodeType}(\text{footer}) \)
16: \( E.\text{addNode}(\text{tempNode}.\text{depth}+1) \)
17: end if
18: \( \text{depth} = \text{depth}+1 \)
19: for all \( \text{node} \in t \) do
20: \( \text{childNodes} = \text{node}.\text{getChildren}() \)
21: for all \( c \in \text{childNodes} \) do
22: if \( c \in t \) then
23: \( E.\text{addNode}(c, \text{depth}+1) \)
24: \( E.\text{setHead}(c) \)
25: else if \( c.\text{status} == \text{complement} \) then
26: \( c.\text{setNodeType}(\text{substitution}) \)
27: \( E.\text{addNode}(c, \text{depth}+1) \)
28: end if
29: end for
30: \( \text{depth} = \text{depth}+1 \)
31: end for
32: return \( E \)
trees. The extracted elementary trees are added to the set `elementaryTrees` (line 9).

Algorithm 2 shows the extraction procedure to extract initial trees. The extraction procedure used to extract auxiliary trees is shown in Algorithm 3. Nodes on the trunk of a parse tree are copied to the elementary structure and marked as the head node for nodes occurring at a level one below the node (lines 8–9). The `addNode` method accepts the child and parent nodes as arguments, and inserts the child node at the end of the existing list of children corresponding to the parent node. The complement nodes attached to the nodes on a trunk are added to the elementary tree and marked as substitution nodes (line 11–12) and the trees they dominate become initial trees.

The adjunct nodes are factored into auxiliary trees such that nodes that are at a distance greater than three from the head node adjoin at the parent node, while those closer adjoin at the head node (lines 5–16). The auxiliary trees extracted through this procedure are `modifier auxiliary trees`. The extraction algorithm for auxiliary trees extracts `predicative auxiliary trees` as well, where the trunks are extended to include nodes with long-range dependencies. While the modifier auxiliary trees modify the constituent they adjoin on to, predicative auxiliary trees subcategorize for the constituent they adjoin to (Bangalore and Joshi, 1995).

In the remainder of this section, we discuss three important issues encountered during grammar extraction.

1. **Head Percolation for discontinuous phrases and coordination constructions**: Discontinuity of constituents in the Spanish treebank is annotated by assigning indexes as features to the nodes dominating the two discontinuous phrases. This involves mainly noun phrases, for which a noun complement is not immediately dominated by a noun phrase because there is another element in between. We collapsed the discontinuous non-terminal nodes to compute the head word for discontinuous constituents, thereby modifying the word order of the original sentence. However, the modified order of constituents is not reflected in the factorization step. This helps us retain the original tree structure assigned in the annotation in the elementary structures, extracting a elementary tree with two nodes indicating the discontinuity rather than a single node for the collapsed non-terminal nodes. Additionally, we chose the head of a coordination construction to be the conjunction element, rather than one of the coordinated elements as done in (Civit et al., 2006).

2. **Determining the status of a node for non-verbal arguments**: As mentioned earlier, the status of arguments of verbal predicates are decided based on the PropBank label assigned to each of the arguments. However in the case non-verbal arguments, function roles have been assigned only to arguments which are directly dominated by a clausal node. For the rest of the arguments (specifier, nominal prepositions, clausal arguments), we use the roles assigned in the dependency representation to determine the status of a node. It has been our observation that these roles alone are not sufficient to determine whether the node is an argument or an adjunct. We use the dependency roles combined with local structural information to determine the status of a node.

3. **Factoring parse trees containing elliptic elements and null elements**: Spanish being a pro-drop language, the treebank contains a large number of null elements in addition to a number of elliptic nodes. Besides, movement in raising structures is not traced in the treebank. As such, the extracted elementary structures do not contain information about traces unlike the structures extracted by (Chen et al., 2006) from the PTB corpus.

### 6 Evaluation

Evaluation of syntactic grammars in previous works has been done by comparing the output of a parser that uses the grammar to a treebank giving a global evaluation to the entire grammar (Briscoe et al., 2002) or by assessing the quality of a part of the grammar for a particular linguistic phenomena (Hockey and Egedi, 1994).

Empirical evaluation of treebank grammars is generally done by comparing against a handcrafted grammar and reporting metrics like coverage and matching. (Xia and Palmer, 1999) report the quality of the grammar extracted from the PTB by comparing it with the hand-crafted XTAG English grammar (XTAG, 2001). We can not
compare the grammar extracted in our work since such a hand-crafted grammar based on the LTAG formalism does not exist for Spanish. Similarly, the methods used in the previous works can not be compared with our method because the earlier methods do not use semantic information which is present in the Spanish treebank and conversely, because the Spanish treebank does not contain the trace information available in PTB and used to determine the argument/adjunct status of a node.

In such a scenario, we resort to the evaluation scheme used in (Hockenmaier, 2006) by reporting the size of the lexicon and the lexical coverage in the extracted grammars. In the current work, we additionally report the performance of a MaxEnt based supertagger on the lexicon extracted from the treebank.

**Datasets:**
The entire treebank was partitioned into three sets to recreate the data released for the CoNLL 2009 shared task (Hajič et al., 2009). The statistics from the datasets are reported in Table 2. The partitions created were intended to be used in the set of parsing experiments, which is a part of our future work. In the following discussion, we report the results on the evaluation set of the partitioned treebank.

**Size of the Grammar:**
The size of the grammar in terms of lexicalized trees and supertags extracted from each dataset are shown in Table 3. The number of lexicalized trees reflects the variation in syntactic contexts found with respect to each lexical item in the treebank.

| Training set | # of token types | 41,057 |
| Development set | # of token types | 10,325 |
| Evaluation set | # of token types | 10,508 |

Table 2: Statistics obtained from the Spanish datasets.

| Training set | # of sentences | 14,058 |
| Development set | # of sentences | 1,623 |
| Evaluation set | # of sentences | 1,665 |

| Training set | # of supertags | 9,212 |
| Development set | # of supertags | 2,292 |
| Evaluation set | # of supertags | 2,378 |

Table 3: Size of the extracted grammar in terms of tree frames and number of lexicalized trees.

On the other hand, the number of supertags indicate the size of the lexicon obtained from the grammar extraction procedure.

**Coverage:**
In order to evaluate the coverage of the extracted lexicon on unseen data, we extracted the lexicons from both the training and the evaluation sets. We found that 73.5% of the lexical tokens that appear in the evaluation set also appear in the training data. We did not find much improvement when the coverage was computed between lemma of both datasets(75.4%).

**Lexicon Coverage:**
We evaluate the extracted grammars in terms of

| % of supertags found in the training set | 73.25% |
| % of supertags not found | 26.75% |
| % lexical anchors of supertags present | 4.62% |
| % lexical anchors of supertags not present | 22.11% |

Table 4: Coverage of extracted grammar in terms of percentage of frames in the evaluation set. Missed coverage is divided into two, lexical anchor seen in the training set and not seen in the training set.

We determined the degree of coverage of a particular grammar by the overlap of elementary trees extracted from the evaluation set with that of the training set. For frames which were not found in the training corpus, we also report the percentage of instances in which the lexical anchors of the supertags were present or not in the training set. Table 4 shows the statistics that were obtained from the evaluation set.

**Supertagging:**
We measure the accuracy of the supertagging models based on the extracted Spanish LTAG grammar. The supertagger estimates the probability of a tag sequence using a discriminative classification approach, in which the context of the word being supertagged is encoded as features for the classifier (Bangalore et al., 2005). The multi-class labeling task of supertagging is modeled using $n$ one-vs-rest binary classifiers (where $n$ is the size of the supertag set). The supertagger uses the lexical, POS attributes from a context window of size 12 (a window of six to the left and six to the right) and the lexical, orthographic and POS attributes of the word being supertagged as features. We train a Maxent model on the training partition of the
treebank and use the evaluation partition for testing the performance of the supertagger. The supertagger achieves an error rate of 20.36% on the evaluation partition, despite the low lexical coverage across both the partitions of the treebank. We found that the supertagger using non-lexical features (part-of-speech tags, orthographic features) achieves a good generalization performance.

7 Conclusions and Future work

In this work, we present a grammar extraction system that extracts an LTAG grammar for Spanish from a treebank. In the absence of a handcrafted Spanish LTAG grammar, we evaluate the extracted grammar for its coverage on the evaluation set. We also present a Spanish supertagger, trained using a maximum entropy model on the extracted LTAG grammar. We evaluate the grammar in terms of lexicon size, coverage and supertagger performance. The lexicon extracted from the Spanish treebank contains entries for 73.25% of the lexical items in the evaluation set. The quality of the extracted grammar is evaluated based on the performance of a supertagger trained on it. The supertagger performs with an error rate of 20.36% and is expected to be deployed for context disambiguation in MT. Given the large percentage of unseen lexical items (30%), the issue of smoothing while estimated the probabilities in the extracted grammar is of crucial importance for improving the coverage of the extracted grammar and therefore, part of our immediate future work. Also, a detailed study of the LTAG grammar extracted for Spanish needs to be taken up in order to ascertain the linguistic coverage. In addition, since the AnCora treebank is annotated with complete morphological and syntactic information, we intend to extract grammars of increasing granularity in the feature-based TAG framework as part of future work.

References


