Semi-supervised Constituent Grammar Induction Based on Text Chunking Information*

Jesús Santamaría and Lourdes Araujo

U. Nacional de Educación a Distancia, NLP-IR Group, Madrid, Spain
{jsant,lurdes}@lsi.uned.es

Abstract. There is a growing interest in unsupervised grammar induction, which does not require syntactic annotations, but provides less accurate results than the supervised approach. Aiming at improving the accuracy of the unsupervised approach, we have resorted to additional information, which can be obtained more easily. Shallow parsing or chunking identifies the sentence constituents (noun phrases, verb phrases, etc.), but without specifying their internal structure. There exist highly accurate systems to perform this task, and thus this information is available even for languages for which large syntactically annotated corpora are lacking. In this work we have investigated how the results of a pattern-based unsupervised grammar induction system improve as data on new kind of phrases are added, leading to a significant improvement in performance. We have analyzed the results for three different languages. We have also shown that the system is able to significantly improve the results of the unsupervised system using the chunks provided by automatic chunkers.

1 Introduction

The aim of Grammar Induction (GI) is to extract a grammar from a collection of texts. There are three main approaches to GI: (1) supervised (SGI), which requires syntactically annotated examples — not available for many languages— (2) unsupervised (UGI), which only requires raw texts[1], and usually achieves a lower performance than the supervised approach, and (3) semi supervised (SSGI), which is an intermediate approach that requires some degree of supervision but not fully syntactically annotated texts. Much research has been developed within the three types of GI. Recent works [9,4,16,8] on SGI report results ranging between 90% and 93% when applied to the WSJ section of the Penn Treebank[2].

Some of the most recent works devoted to the UGI approach are due to Klein and Manning [5], Bod [2], Seginer [12] and Santamaria and Araujo [11]. The UGI system by Santamaria and Araujo [11] is based on the statistical detection of certain POS tag patterns that appear in the sentences. The goal is to find a POS tag classification according to the role that the POS tag plays in the parse trees. This system, which considers

* Supported by projects Holopedia (TIN2010-21128-C02) and MA2VICMR (S2009/TIC-1542).

[1] It is also considered an unsupervised approach if you use the Part-of-Speech (POS) tags associated with the words of the sentence.

[2] The Wall Street Journal section of Penn Treebank corpus [6].

A. Gelbukh (Ed.): CICLing 2013, Part I, LNCS 7816, pp. 258–269, 2013.

any kind of tree — not only binary ones— has achieved an F-measure of 75.82%, without including constituents spanning the whole sentence, and 80.50% including them, as the other works do.

Finally, there have also been an increasing interest on the semi supervised approach (SSGI), though most works in this line are devoted to dependency parsing. Klein and Manning have also tried to include some supervision in the work mentioned previously [5]. Haghighi and Klein [3] have also proposed a model to incorporate knowledge to the CCM model by Klein and Manning [5]. In this case, prior knowledge is specified declaratively, by providing a few canonical examples of each target phrase type.

The supervised approach requires a syntactically annotated corpus which is not available for many languages. Moreover, current treebanks are usually composed of a particular kind of text (usually newspaper articles) and may not be appropriate for others. This makes the unsupervised approach worth considering. In order to bridge the performance gap between supervised and unsupervised systems we propose to introduce a very low degree of supervision which could be achieved without requiring fully syntactic annotation of texts. Accordingly, we have resorted to the so-called *shallow parsers*, which aim at identifying the constituents of a sentence, but do not specify their internal structure. In the experiments carried out in this work we have first analyzed the results obtained using the chunks annotated in a treebank for English. We have later evaluated the system using automatic chunkers and other languages. Our proposal is considered a semi-supervised approach, since it relies either on the chunk annotations or on an accurately trained chunker. Several chunkers reports results above 95% [10,13,14] and even higher for some particular type of phrase, such as NPs [1], for which a precision of 97.8% is achieved. Accordingly, we expect that the results would only be slightly affected by using a chunker instead of chunking annotations. Certainly the high performance mentioned chunkers are supervised systems, but the amount of data needed, for instance, to train a finite state automaton for NP detection, VP detection, etc. is much smaller than the one required for supervised parsing.

We have tested the effect of adding different degrees of supervision to the underlying UGI system. Specifically we have checked how the performance is affected as the *chunks* corresponding to different type of phrases —noun, verb and prepositional phrases— are identified prior to the parsing process. We have chosen the Santamaria and Araujo model [11] as the underlying UGI system because it extracts patterns which can be easily combined with others representing the introduced supervision — it is not restricted to binary trees— and at the same time provides competitive results. The addition of this knowledge has improved the performance of the underlying UGI system from 75.82% to 81.51% with the lower level of supervision, thus proving the usefulness of the proposal. However, the approach described in this work could also be applied to other UGI systems.

The paper is organized as follows: section 2 briefly describes the underlying UGI system, Section 3 explains two strategies to introduce the chunking information to the model, that thus becomes a SSGI system, Section 4 discusses the results, and finally the conclusions are outlined in Section 5.

2 Pattern Based Unsupervised Grammar Induction

The underlying system [11] for UGI relies on the idea that the different POS tags play particular roles in the parse trees. Thus, the POS tags can be classified in a small number of POS tag classes, whose elements have a tendency to appear at certain positions of the parse tree. Furthermore, this feature is shared by many languages. This proposal applies some heuristics based on the frequencies of the POS tag sequences to classify the POS tag set of a corpus into POS tag classes. This classification is then used to parse any sentence by means of a deterministic and fast procedure which selects the constituents and the branching of the parse tree depending on the POS tag classes of the tags in the sentence.

Let us consider a parse tree example to illustrate the different types of POS tag classes considered in the model. Examining the example appearing in Figure 1 we can observe that the POS tags are distributed in different levels which divide the POS tag sequence. Some POS tag sequences tend to appear at the boundaries of the constituents. This is the case of (DT, NNP) and (DT, NN), where DT is a determiner, NNP a proper singular noun, and NN a singular noun. Other POS tags such VBZ (verb, 3rd ps. sing. present) and TO (To) give rise to a new level in the parse tree since a new subtree appears after them. The proposed model tries to capture these ideas, identifying a number of POS tag classes which correspond to particular structural patterns.

The procedure to identify the different classes of POS tags relies on the detection of the minimum constituent (MC):

The MC is the most frequent sequence of two POS tags in the corpus.

It is expected that at least for the most frequent constituent the number of occurrences overwhelms the number of sequences appearing by chance. In the WSJ10 corpus the MC is DT NN*, where NN* represents any type of noun (NN, NNS, NNP, NNPS).

The model considers the following set of POS tag classes, whose detection is based on the POS tag behavior with regard to the MC:

- *Expanders*: POS tags that have a tendency to appear inside the MC. This is often the case of the tag JJ (adjective) which appears between a determiner (DT) and a noun (NN).
- *Separators*: POS tags with a statistical tendency to appear outside the MC. This kind of POS tag usually gives rise to a new level in the parse tree. This is the case of VBZ (verb, 3rd person singular present) in the parse tree of Figure 1.
- *Subseparators*: POS tags which do not have a tendency to be inside or outside the MC. They usually appear delimiting constituents, but without giving rise to new levels in the parse tree as the separators do.

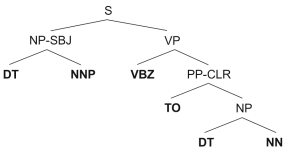

Fig. 1. Parse tree for the sentence *The U.S.S.R. belongs to neither organization*

A POS tag is identified as an expander, a separators or subseparators by computing frequencies of occurrence at each side of the boundaries of the CM. A detailed description of the process can be found in [11].

The set of separators obtained applying the described procedure to the WSJ10 corpus is (MD, PRP, IN, RB, RBR, CC, TO, VB, VBD, VBN, VBZ, VBP,VBG, EX, LS, RP, UH, WP, WRB, WDT), and the set of subseparators is (DT, PDT, POS, SYM, NN, NNS, NNP, NNPS). Sub-separators can be grouped to their right or to their left. The bias of each of them for one direction or another is determined by comparing the number of occurrences of the most frequent sequence composed of the sub-separator and a POS tag on the right and on the left. Then, the preference direction for a sub-separator is the one corresponding to the most frequent sequence. According to this criteria DT, PDT, SYM tend to appear on the left of the constituent, whereas POS, NN, NNS, NNP and NNPS on the right.

The classification of the POS tag set in separators, sub-separators and expanders allow defining a deterministic parsing procedure based on the behavior expected for each POS tag in the sentence. The steps of this procedure are the following:

- Identify the separators. In the sentence example *The (DT) computers(NNS) were(VBD) crude(JJ) by(IN) today(NN) 's(POS) standards(NNS)* separators appear in boldface:

  [DT NNS **VBD** JJ **IN** NN POS NNS]

- Split the sentence according to the separators. The first separator which is a verb, if any, is used to split the sentence into two parts.

  [[DT NNS] [**VBD** [JJ [**IN** [NN POS NNS]]]]]

  Each separator can give rise to two groups: one composed of the tag sequence between the separator and the next separator, and another one which includes the separator and the POS tags up to the end of the part of the sentence in which it appears.
- Identify the subseparators, underlined in the sentence:

  [[DT NNS] [**VBD** [JJ [**IN** [NN POS NNS]]]]]

- Split the sentence according to the sub-separators forming groups according to their preference direction.

  [[DT NNS] [**VBD** [JJ [**IN** [[NN POS] NNS]]]]]

At this point the parse tree for the sentence has been obtained.

3 Introducing Text Chunking Information

Text chunking is a partial parsing which amounts to dividing sentences into nonoverlapping segments. Chunkers assign a partial syntactic structure to a sentence, yielding flatter structures than full parsing. They usually provide chunks for a fixed tree depth, instead of arbitrarily deep trees.

The SSGI approach we have adopted assumes that the chunks corresponding to some particular types of phrases and tree depths have been prior identified. We have designed two different ways of applying the identified chunks. The first one, called *constraining strategy*, applies chunks in advance to the parsing process, as a kind of constraint. The second one, called *error-correcting strategy*, generates the parse tree applying the UGI system and then uses the chunks to correct some possible errors. In the next subsections we described each of them.

3.1 Constraining Strategy

The idea underlying this strategy is to apply the UGI system to infer the different subtrees of the parse tree resulting from the chunks identification. One of these subtrees is the top subtree obtained by substituting the identified chunks by special tags representing some of the POS tag classes. The other subtrees are the ones corresponding to each identified chunk. Specifically, we perform the following process:

1. Extract the constituents of the kind (noun phrases, verb phrases, etc.) and level under consideration from the parse tree.
2. Replace in the sentence the constituents extracted in the previous step by special POS tags which belong to some of the POS tag classes considered in the UGI system. Then, apply the UGI system to generate the tree for the new sentence.
3. Apply the UGI system to build the sub-tree for the constituents extracted in step 1.
4. Replace the special POS tags of the tree generated in step 2, for the sub-trees generated in step 3.

The result of this process depends on the types of constituents being identified in advance. In order to clarify the described procedure let us see an example. Assume that we consider only noun phrases (NP) as the type of constituent being extracted in advance. Then NPs have to be substituted by a special POS tag which is being assigned to one of the POS tag classes of the UGI system: separators, subseparators and expanders. In the case of NPs we have chosen the subseparator class since the head of this kind of phrase, the nouns (NN, NNS, NNP and NNPS), have been identified as subseparators by the UGI system. Accordingly we have defined the special POS tag SUBS to substitute the NPs. Now, we apply the described procedure to the sentence appearing in Figure 2(a), considering only top level NPs:

1. The top level NPs are identified: NP = [DT NN IN PRP$ NN NN NN].
2. The NP identified in the previous step is replaced by SUBS, giving rise to the tree appearing in Figure 2(b).
3. The UGI system is also applied to generate the parse tree for the replaced NP. The result appears in Figure 2(c). Notice that we are not assuming to have information on the deep parsing of these constituents, but only on the chunking process, which is a much more weak requirement.
4. Finally, the special POS tag SUBS is replaced by the parse tree found for the constituent in step 3, obtaining the tree shown in Figure 2(d).

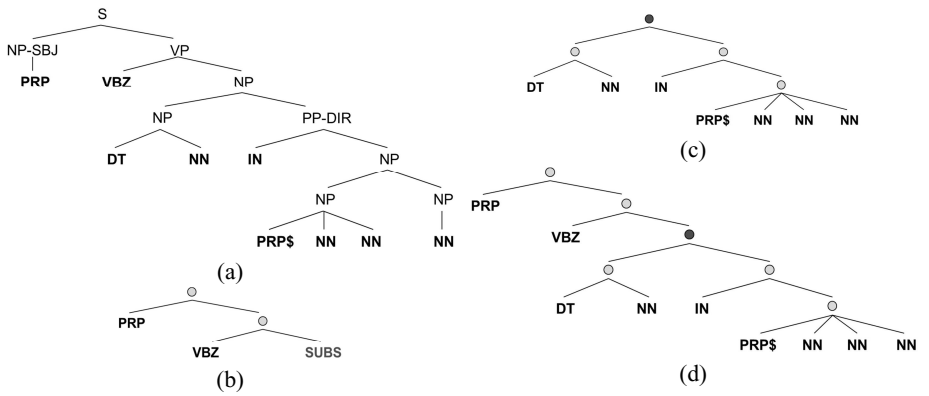

Fig. 2. (a)Parse tree for the sentence *It has no bearing on our work force today*. from the Wall Street journal part from the Penn treebank. (b) Tree provided by the UGI system for the sequence of POS tags obtained after replacing the top level NP in the tree in Figure (a) by the special POS tag SUBS. (c) Tree generates by the UGI system for the sequence of POS tags corresponding to the NP replaced in Figure (a). (d) Final tree obtained by the semisupervised GI system for the sentence in Figure (a).

For other types of phrases the process is analogous. In the case of verb and prepositional phrases the replacing special POS tag is called SEP, which is considered part of the separator class. The reason is that the head of the verb phrases, the verb (VB, VBD, VBN, VBZ, etc.), belongs to the separator class. We can also observe that the preposition (IN), the head of the prepositional phrases, also belongs to this class.

3.2 Error-Correcting Strategy

This strategy applies the UGI system to generate the initial parse tree for a sentence. Then, the identified chunks are applied to correct possible errors in the generated parse tree. The gold chunks are compared to each constituent in the UGI tree sharing some word (POS tag corresponding to the word). The UGI constituent is considered wrong if there is a partial overlap between it and the gold chunk. In this case the constituent containing the overlapping sequence of POS tags is removed and the overlapping sequence is considered a constituent. Once all the conflicts have been solved, the gold chunks are included as constituents, provided they had not been identified by the UGI system. Let us consider the following sentence *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29*, from the Penn Treebank. According to the gold standard the higher level VP for this sentence is:

MD	VB	DT	NN	IN	DT	JJ	NN	NNP	CD
6	7	8	9	10	11	12	13	14	15

where the number below the POS tags represents the position of the corresponding word in the sentence. Let us assume that the parse tree generated by the UGI system is the one appearing in Figure 3(a). This tree presents the following constituent at the beginning of the sentence:

NNP NNP CD NNS JJ MD
1 2 3 4 5 6

We can observe that there is a conflict with the POS tag MD at position 6, since it belongs to two different constituents. Then MD is included as a new constituent, and [NNP NNP CD NNS JJ MD] is not considered a constituent any more. Notice that there is not conflict between the gold VP chunk and the UGI constituent

VB DT NN IN DT JJ NN NNP CD
7 8 9 10 11 12 13 14 15

because the latter is completely included in the chunk. Figure 3(b) presents the final tree obtained by applying the gold chunks.

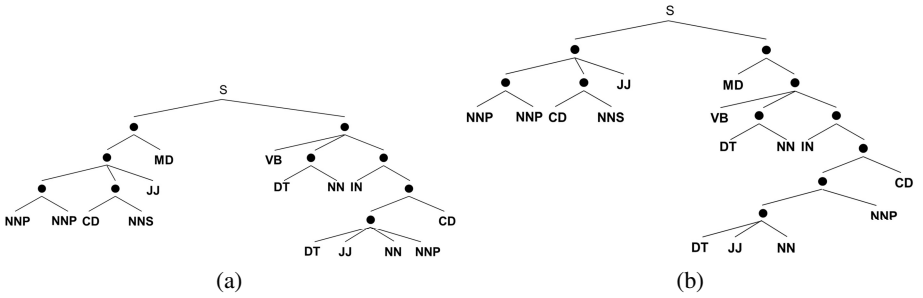

Fig. 3. (a) Tree obtained by the UGI system for the sentence *Pierre Vinken, 61 years old, will join the board as a nonexecutive director Nov. 29.. (b) Final tree obtained after correcting the errors in the UGI tree of Figure (a) by applying the gold chunks.*

This approach can take better advantage of the result of the UGI system, because it is applied to the whole sentence, instead of to short segments of it, as the constraining approach does.

4 Experimental Results

We have used the WSJ10 corpus, with 7422 sentences, in our first experiments. It is composed of the sentences of up to ten words from the Wall Street Journal section from the Penn Treebank. Though this corpus is syntactically annotated, we have used this information only to obtain the chunks that are expected to be provided by a shallow parser, thus not relying on the performance of a particular system, and for evaluation purposes. In order to evaluate the quality of the obtained grammar we have used the most common measures for parsing and grammar induction evaluation: recall, precision, and their harmonic mean (F-measure).

The underlying UGI system achieves an UF of 75.82%, which is used as our baseline. Next, we discuss the results obtained when a perfect shallow parser[3] provides us

[3] At this point we use the treebank annotations for the chunks corresponding to the different kind of phrase.

the boundaries of the noun phrases, verb phrases and prepositional phrases. We have studied the effect of using information on the chunking corresponding to each kind of constituent separately, before evaluating the results of applying all of them simultaneously. For each of them we have also investigated two different levels of supervision. In subsection 4.3 we present results obtained for different languages. Finally, subsection 4.4 shows results obtained using automatic chunkers.

4.1 Isolated Effect of Some Kind of Phrases

Table 1(a) compares, for different kind of phrases and supervision levels, the UF achieved by the two strategies, constraining (SSGI_Cons) and error-correcting (SSGI_EC) of the SSGI system, and the baseline given by the UGI system. In considering the different types of phrase separately, supervision levels refer only to tree depths in which the type of phrase under consideration appears. The two first rows show the results using the chunks corresponding to the NPs appearing in the sentence at level 1 and 2, respectively. In the case of the constraining strategy, NP chunks are replaced by the special SUBS POS tag, which behaves as a subseparator. The supervision introduced by the NP chunking significantly improves the results. Even for the lower supervision level, i.e., considering only the chunks corresponding to the minimum depth NPs appearing in the sentence (at a supervision degree of 17,55%) we obtain an improvement of 3,12% over the UGI baseline. The two considered strategies significantly improve the results for all levels, though the error-correcting one is slightly better. We think that the reason may be that with the error correcting strategy the UGI system is applied to the whole sentence, taking advantage of more information that the constraining strategy.

The next two rows in Table 1(a) show the results for VPs annotated at level 1 and 2, respectively. In the case of the constraining strategy, VP chunks are replaced by the special SEP POS tag, which behaves as a separator. Again the error-correcting strategy performs better. The improvement achieved in this case is higher than for NPs. The reason may be that the verb phrases are more difficult to be captured in an unsupervised manner, and therefore the VP information is more helpful. This information also helps to know the point that separates the subject and the predicate. For the lower supervision level, i.e., considering only the chunks corresponding to the minimum depth VPs appearing in the sentence (at a supervision degree of 21,81%) we obtain an improvement of 6,11% over the UGI baseline.

The last two rows in Table 1(a) show the results for PPs annotated at level 1 and 2, respectively. In the case of the constraining strategy, PP chunks are replaced by the special SEP POS tag, which behaves as a separator. Once again the error-correcting strategy performs better. The rate of improvement in this case is lower, as well as the supervision degree, indicating that PPs appear less often. Considering only the chunks corresponding to the minimum depth PPs appearing in the sentence (at a very low supervision degree of 5,51%) we obtain an improvement of 1.59% over the UGI baseline.

4.2 Combined Effect

We can consider different kinds of phrases annotated simultaneously. Table 1(b) shows the results obtained the first two supervision levels. The error-correcting strategy is

Table 1. Applying chunking for some kind of phrases in WSJ10: F-measure corresponding to the baseline (UGI) and the SSGI systems, the constraining strategy (SSGI_Cons) and the error-correcting strategy (SSGI_EC) per supervision level, (a) considering each kind of phrase and (b) all together. The number in bracket next to the SSGI results indicates the improvement rate with respect to the UGI F-measure.

P. kind	Superv. Level	UGI	SSGI_ Cons	SSGI _EC
NP	1	75,82	77,67(2,43)	78,19(3,12)
NP	2	75,82	78,85(3,99)	79,07(4,28)
VP	1	75,82	79,87(5,34)	80,46(6,11)
VP	2	75,82	80,19(5,76)	80,77(6,52)
PP	1	75,82	76,44(0,81)	77,03(1,59)
PP	2	75,82	76,47(0,85)	77,05(1,62)

(a)

Superv. Level	UGI	SSGI_ Cons	SSGI _EC
1(26,40%)	75,82	81,18(7,06)	81,51(7,50)
2(47,93%)	75,82	83,31(9,87)	83,66(10,34)

(b)

Table 2. Results for WSJ10 (a) and WSJ30(b), considering only minimum depth chunks, of unlabeled recall (UR), precision (UP), and F-measure (UF) for the baseline provided by the underlying UGI system, the semi-supervised system (error-correcting strategy) which uses the NPs chunks (SSGI-NP), the one which uses the VP chunks (SSGI-VP), the one which uses the PP chunks (SSGI-PP) and the whole semisupervised system (SSGI).

System	UR	UP	UF(% Improv.)
UGI	77.99	73.77	75.82
SSGI-NP	81.44	75.18	78.19(3.12%)
SSGI-VP	82.89	78.17	80.46(6.11%)
SSGI-PP	79.46	74.74	77.03(1.59%)
SSGI	84.28	78.92	81.51(7.50%)

(a)

System	UR	UP	UF(% Improv.)
UGI	61.35	64.07	62.68
SSGI-NP	70.59	70.09	70.34(12.22%)
SSGI-VP	68.68	71.26	69.95(11.59%)
SSGI-PP	66.21	67.57	66.88(6.70%)
SSGI	71.23	71.43	71.33(13.80%)

(b)

again the better. The combined effect of identifying NPs, VPs, and PPs in advance improves the effect obtained for each of them separately. The results at lowest level of supervision, that can also be achieved by automatic chunker, are particularly interesting. Table 2(a) shows the results (recall, precision and F-measure, as long as the rate of improvement achieved for the last one) for the error-correcting strategy of the different systems considered, and for the minimum tree depth in which the phrases under consideration appear. We can observe that all proposed systems improve the three measures, being the F-measure improvement achieved by the final system more than 7%. Of course, the rate of improvement is lower than the degree of supervision introduced because many constituents had also been identified by the UGI system.

We have also investigated how our SSGI system behaves for longer sentences. We have considered the WSJ30 corpus, composed of 41227 sentences of up to 30 words from the WSJ part of the Penn Treebank. Table 2(b) shows the results for the error-correcting approach and for the minimum depth chunks. We can see that for longer sentences the improvement is larger, leading to high performance results even for the

smaller level of supervision (22.21% for the SSGI). Another interesting observation is that for longer sentences the NP supervision becomes more relevant. As sentences get longer, NPs are more complex and difficult to detect in an unsupervised way.

4.3 Other Languages

We have also evaluated the semisupervised system for other languages, specifically, German and Spanish. We have used the UAM Spanish Treebank [7], with 1501 syntactically annotated sentences, and the German corpus NEGRA [15], with 20,602 sentences from German newspaper texts. For comparison, we have used a restricted part of the corpora with sentences of up to 10 words, which we call UAM10, and NEGRA10. Table 3(a) shows the results obtained using NPs, VPs and PPs at two different supervision levels. We can observe that the two proposed approaches are able to obtained a significant performance improvement even for the minimum supervision level and in both languages. These results prove the viability of the method on different languages.

Table 3. (a) Results for Spanish (UAM10) and German (NEGRA10) using NP, VP and PP annotations for different levels of supervision. (b) Results for automatic chunkers. The first column indicates the languages considered: English (Engl.), Spanish (Span.) and German (Germ.). The second column shows the unsupervised system results for comparison purposes. The third and fourth columns show the results of the semi-supervised system using the chunk annotations provided by automatic chunkers (Treetagger for English and German, and Freeling for Spanish). The last two columns correspond to the results obtained with the two proposed approaches using the treebank annotations, also for comparison purposes.

Lang.	Superv. Level	UGI	SSGI_ Cons	SSGI _EC
Spanish	1(32.7)	70.68	75.78	76.03
Spanish	2(62.9)	70.68	80.35	81.07
German	1(60.2)	49.67	56.98	55.87
German	2(75.9)	49.67	59.01	58.85

(a)

		Tagger		**TB annot.**	
Lang.	UGI	SSGI_ Cons	SSGI _EC	SSGI_ Cons	SSGI _EC
Engl.	75.82	72.17	72.10	81.18	81.51
Span.	70.68	75.05	74.21	75.78	76.03
Germ.	49.67	52.77	53.52	56.98	55.87

(b)

4.4 Experiments with Automatic Chunking

Finally, we provide evaluation results for automatically identified chunk boundaries. We have used the Treetagger chunker [4] for English and German, and the Freeling chunker [5] for Spanish. Table 3(b) shows the results obtained for the minimum supervision level, i.e., without using nested chunks. We can observe a significant improvement over the unsupervised results (first column) for Spanish and German. This improvement is slightly lower than the one obtained using the treebank annotations (two last columns), but is close to it for both languages. The English case is different. The chunker extracts "bare NPs" without postmodifiers and verb groups (VGs not VPs) without complements. These constituents do not match those of the Penn Treebank, and thus the

[4] `http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/`.
[5] `http://www.lsi.upc.edu/~nlp/freeling`.

semisupervised system worsens the unsupervised results for English. However, this is simply a mismatch in the annotation schemes of the chunker and the treebank used for evaluation, and not an inherent problem of the proposed approaches, as the results for other languages show. The results obtained for Spanish and German show the viability of the proposed approaches to take advantage of current automatic chunkers.

5 Conclusions

We have presented a semi supervised method for grammar inference. The proposed model extends an unsupervised GI model based on the detection of POS tag classes classified according to the tag role in the structure of the parse trees. The supervision added, which amounts to knowing the *chunks* corresponding to the some types of phrase (noun, verb and prepositional phrases) at a certain tree depth, is introduced in the unsupervised system in a very natural way following two different strategies. The error-correcting strategy, which uses the chunks to correct possible errors in the tree generated by the UGI system, has proven to be slightly better. We have investigated the effect of knowing in advance the NPs, VPs, and PPs. Results obtained by this semi-supervised system using gold chunks improve those obtained by the underlying UGI system in all cases. The constituents that have exhibited the greater effect on the results are VPs, showing the difficulty of detecting this kind of constituent in an unsupervised manner. However, all of them have improved the results, even for the lowest supervision degree. Furthermore, results improve with the length of the sentences. We have tested the system for English, Spanish and German, significantly improving the unsupervised system performance for all of them. We have been able to improve the performance of the UGI system with the supervision provided by some of the automatic chunkers that currently exist. In fact, we have observed significant improvement using automatic chunkers for Spanish and German, for which the chunker annotation scheme is similar to the one of the treebank used in the evaluation.

References

1. Araujo, L., Serrano, J.I.: Highly accurate error-driven method for noun phrase detection. Pattern Recognition Letters 29(4), 547–557 (2008)
2. Bod, R.: Unsupervised parsing with u-dop. In: CoNLL-X 2006: Proceedings of the Tenth Conference on Computational Natural Language Learning, pp. 85–92. Association for Computational Linguistics, Morristown (2006)
3. Haghighi, A., Klein, D.: Prototype-driven grammar induction. In: Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, pp. 881–888. Association for Computational Linguistics (2006)
4. Huang, L.: Forest reranking: Discriminative parsing with non-local features. In: Proc. of ACL (2008)
5. Klein, D., Manning, C.D.: Natural language grammar induction with a generative constituent-context model. Pattern Recognition 38(9), 1407–1419 (2005)
6. Marcus, M.P., Santorini, B., Marcinkiewicz, M.A.: Building a large annotated corpus of english: The penn treebank. Computational Linguistics 19(2), 313–330 (1994), `citeseer.ist.psu.edu/marcus93building.html`

7. Moreno, A., Grishman, R., Lopez, S., Sanchez, F., Sekine, S.: A treebank of spanish and its application to parsing. In: LREC 2000: Proceedings of the International Conference on Language Resources & Evaluation, pp. 107–111. European Language Resources Association, ELRA (2000)
8. Petrov, S.: Products of random latent variable grammars. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics, HLT 2010, pp. 19–27 (2010)
9. Petrov, S., Klein, D.: Improved inference for unlexicalized parsing. In: HLT-NAACL, pp. 404–411 (2007)
10. Sang, E.F.T.K.: Memory-Based Shallow Parsing. ArXiv Computer Science e-prints (April 2002)
11. Santamaría, J., Araujo, L.: Identifying patterns for unsupervised grammar induction. In: Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pp. 38–45. Association for Computational Linguistics, Uppsala (2010), http://www.aclweb.org/anthology/W10-2905
12. Seginer, Y.: Fast unsupervised incremental parsing. In: Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics, pp. 384–391 (2007)
13. Sha, F., Pereira, F.: Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL 2003, pp. 213–220 (2003)
14. Shen, H., Sarkar, A.: Voting Between Multiple Data Representations for Text Chunking. In: Kégl, B., Lee, H.-H. (eds.) AI 2005. LNCS (LNAI), vol. 3501, pp. 389–400. Springer, Heidelberg (2005)
15. Skut, W., Krenn, B., Brants, T., Uszkoreit, H.: An annotation scheme for free word order languages. In: Proceedings of the Fifth Conference on Applied Natural Language Processing, ANLC 1997, pp. 88–95. Association for Computational Linguistics (1997)
16. Zhang, H., Zhang, M., Tan, C.L., Li, H.: K-best combination of syntactic parsers. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP 2009, vol. 3, pp. 1552–1560 (2009)