Enforcing Subcategorization Constraints in a Parser Using Sub-parses Recombining

Seyed Abolghasem Mirroshandel[†,⋆] **Alexis Nasr**[†] **Benoît Sagot**[◇]
[†]Laboratoire d'Informatique Fondamentale de Marseille- CNRS - UMR 7279
Université Aix-Marseille, Marseille, France
[◇]Alpage, INRIA & Université Paris-Diderot, Paris, France
[⋆]Computer Engineering Department, Faculty of Engineering,
University of Guilan, Rasht, Iran
(ghasem.mirroshandel@lif.univ-mrs.fr, alexis.nasr@lif.univ-mrs.fr,
benoit.sagot@inria.fr)

Abstract

Treebanks are not large enough to adequately model subcategorization frames of predicative lexemes, which is an important source of lexico-syntactic constraints for parsing. As a consequence, parsers trained on such treebanks usually make mistakes when selecting the arguments of predicative lexemes. In this paper, we propose an original way to correct subcategorization errors by combining subparses of a sentence S that appear in the list of the n-best parses of S. The subcategorization information comes from three different resources, the first one is extracted from a treebank, the second one is computed on a large corpora and the third one is an existing syntactic lexicon. Experiments on the French Treebank showed a 15.24% reduction of erroneous subcategorization frames (SF) selections for verbs as well as a relative decrease of the error rate of 4% Labeled Accuracy Score on the state of the art parser on this treebank.

1 Introduction

Automatic syntactic parsing of natural languages has witnessed many important changes in the last fifteen years. Among these changes, two have modified the nature of the task itself. The first one is the availability of treebanks such as the Penn Treebank (Marcus et al., 1993) or the French Treebank (Abeillé et al., 2003), which have been used in the parsing community to train stochastic parsers, such as (Collins, 1997; Petrov and Klein, 2008). Such work remained rooted in the classical language theoretic tradition of parsing, generally based on variants of generative context free grammars. The second change occurred with the use of discriminative machine learning techniques, first to rerank the output of a stochastic parser (Collins, 2000; Charniak and Johnson, 2005) and then in the parser itself (Ratnaparkhi, 1999; Nivre et al., 2007; McDonald et al., 2005a). Such parsers clearly depart from classical parsers in the sense that they do not rely anymore on a generative grammar: given a sentence S, all possible parses for S[1] are considered as possible parses of S. A parse tree is seen as a set of lexico-syntactic features which are associated to weights. The score of a parse is computed as the sum of the weights of its features.

This new generation of parsers allows to reach high accuracy but possess their own limitations. We will focus in this paper on one kind of weakness of such parser which is their inability to properly take into account subcategorization frames (SF) of predicative lexemes[2], an important source of lexico-syntactic constraints. The proper treatment of SF is actually confronted to two kinds of problems: (1) the acquisition of correct SF for verbs and (2) the integration of such constraints in the parser.

The first problem is a consequence of the use of treebanks for training parsers. Such treebanks are composed of a few thousands sentences and only a small subpart of acceptable SF for a verb actually

[1]Another important aspect of the new parsing paradigm is the use of dependency trees as a means to represent syntactic structure. In dependency syntax, the number of possible syntactic trees associated to a sentence is bounded, and only depends on the length of the sentence, which is not the case with syntagmatic derivation trees.

[2]We will concentrate in this paper on verbal SF.

Proceedings of NAACL-HLT 2013, pages 239–247,
Atlanta, Georgia, 9–14 June 2013.

occur in the treebank.

The second problem is a consequence of the parsing models. For algorithmic complexity as well as data sparseness reasons, the parser only considers lexico-syntactic configurations of limited domain of locality (in the parser used in the current work, this domain of locality is limited to configurations made of one or two dependencies). As described in more details in section 2, SF often exceed in scope such domains of locality and are therefore not easy to integrate in the parser. A popular method for introducing higher order constraints in a parser consist in reranking the n best output of a parser as in (Collins, 2000; Charniak and Johnson, 2005). The reranker search space is restricted by the output of the parser and high order features can be used. One drawback of the reranking approach is that correct SF for the predicates of a sentence can actually appear in different parse trees. Selecting complete trees can therefore lead to sub-optimal solutions. The method proposed in this paper merges parts of different trees that appear in an n best list in order to build a new parse.

Taking into account SF in a parser has been a major issue in the design of syntactic formalisms in the eighties and nineties. Unification grammars, such as Lexical Functional Grammars (Bresnan, 1982), Generalized Phrase Structure Grammars (Gazdar et al., 1985) and Head-driven Phrase Structure Grammars (Pollard and Sag, 1994), made SF part of the grammar. Tree Adjoining Grammars (Joshi et al., 1975) proposed to extend the domain of locality of Context Free Grammars partly in order to be able to represent SF in a generative grammar. More recently, (Collins, 1997) proposed a way to introduce SF in a probabilistic context free grammar and (Arun and Keller, 2005) used the same technique for French. (Carroll et al., 1998), used subcategorization probabilities for ranking trees generated by unification-based phrasal grammar and (Zeman, 2002) showed that using frame frequency in a dependency parser can lead to a significant improvement of the performance of the parser.

The main novelties of the work presented here is (1) the way a new parse is built by combining subparses that appear in the n best parse list and (2) the use of three very different resources that list the possible SF for verbs.

The organization of the paper is the following: in section 2, we will briefly describe the parsing model that we will be using for this work and give accuracy results on a French corpus. Section 3 will describe three different resources that we have been using to correct SF errors made by the parser and give coverage results for these resources on a development corpus. Section 4 will propose three different ways to take into account, in the parser, the resources described in section 3 and give accuracy results. Section 5 concludes the paper.

2 The Parser

The parser used in this work is the second order graph based parser (McDonald et al., 2005b) implementation of (Bohnet, 2010). The parser was trained on the French Treebank (Abeillé et al., 2003) which was transformed into dependency trees by (Candito et al., 2009). The size of the treebank and its decomposition into train, development and test sets are represented in table 1.

	nb of sentences	nb of tokens
TRAIN	9 881	278 083
DEV	1 239	36 508
TEST	1 235	36 340

Table 1: Size and decomposition of the French Treebank

The parser gave state of the art results for parsing of French, reported in table 2. Table 2 reports the standard Labeled Accuracy Score (LAS) and Unlabeled Accuracy Score (UAS) which is the ratio of correct labeled (for LAS) or unlabeled (for UAS) dependencies in a sentence. We also defined a more specific measure: the SF Accuracy Score (SAS) which is the ratio of verb occurrences that have been paired with the correct SF by the parser. We have introduced this quantity in order to measure more accurately the impact of the methods described in this paper on the selection of a SF for the verbs of a sentence.

	TEST	DEV
SAS	80.84	79.88
LAS	88.88	88.53
UAS	90.71	90.37

Table 2: Subcategorization Frame Accuracy, Labeled and Unlabeled Accuracy Score on TEST and DEV.

We have chosen a second order graph parser in this work for two reasons. The first is that it is the parsing model that obtained the best results on the French Treebank. The second is that it allows us to impose structural constraints in the solution of the parser, as described in (Mirroshandel and Nasr, 2011), a feature that will reveal itself precious when enforcing SF in the parser output.

3 The Resources

Three resources have been used in this work in order to correct SF errors. The first one has been extracted from a treebank, the second has been extracted from an automatically parsed corpus that is several order of magnitude bigger than the treebank. The third one has been extracted from an existing lexico-syntactic resource. The three resources are respectively described in sections 3.2, 3.3 and 3.4. Before describing the resources, we describe in details, in section 3.1 our definition of SF. In section 3.5, we evaluate the coverage of these resources on the DEV corpus. Coverage is an important characteristic of a resource: in case of an SF error made by the parser, if the correct SF that should be associated to a verb, in a sentence, does not appear in the resource, it will be impossible to correct the error.

3.1 Subcat Frames Description

In this work, a SF is defined as a couple (G, L) where G is the part of speech tag of the element that licenses the SF. This part of speech tag can either be a verb in infinitive form (VINF), a past participle (VPP), a finite tense verb (V) or a present participle (VPR). L is a set of couples (f, c) where f is a syntactic function tag chosen among a set $\mathcal{F}$ and c is a part of speech tag chosen among the set $\mathcal{C}$. Couple (f, c) indicates that function f can be realized as part of speech tag c. Sets $\mathcal{F}$ and $\mathcal{C}$ are respectively displayed in top and bottom tables of figure 1. An anchored SF (ASF) is a couple (v, S) where v is a verb lemma and S is a SF, as described above.

A resource is defined as a collection of ASF (v, S), each associated to a count c, to represent the fact that verb v has been seen with SF S c times. In the case of the resource extracted form an existing lexicon (section 3.4), the notion of count is not applicable and we will consider that it is always equal to one.

SUJ	subject
OBJ	object
A_OBJ	indirect object introduced by the preposition à
DE_OBJ	indirect object introduced by the preposition de
P_OBJ	indirect object introduced by another preposition
ATS	attribute of the subject
ATO	attribute of the direct object

ADJ	adjective
CS	subordinating conjunction
N	noun
V	verb finite tense
VINF	verb infinitive form
VPP	verb past participle
VPR	verb present participle

Figure 1: Syntactic functions of the arguments of the SF (top table). Part of speech tags of the arguments of the SF (bottom table)

Below is an example of three ASF for the french verb *donner* (*to give*). The first one is a transitive SF where both the subject and the object are realized as nouns as in *Jean donne un livre* (*Jean gives a book.*). The second one is ditransitive, it has both a direct object and an indirect one introduced by the preposition *à* as in *Jean donne un livre à Marie.* (*Jean gives a book to Marie*). The third one corresponds to a passive form as in *le livre est donné à Marie par Jean* (*The book is given to Marie by Jean*).

```
(donner,(V,(suj,N),(obj,N)))
(donner,(V,(suj,N),(obj,N),(a_obj,N)))
(donner,(VPP,(suj,N),(aux_pass,V),
             (a_obj,N),(p_obj,N)))
```

One can note that when an argument corresponds to an indirect dependent of the verb (introduced either by a preposition or a subordinating conjunction), we do not represent in the SF, the category of the element that introduces the argument, but the category of the argument itself, a noun or a verb.

Two important choices have to be made when defining SF. The first one concerns the dependents of the predicative element that are in the SF (argument/adjunct distinction) and the second is the level of abstraction at which SF are defined.

In our case, the first choice is constrained by the treebank annotation guidelines. The FTB distinguishes seven syntactic functions which can be considered as arguments of a verb. They are listed in the top table of figure 1. Most of them are straight-

forward and do not deserve an explanation. Something has to be said though on the syntactic function P_OBJ which is used to model arguments of the verb introduced by a preposition that is neither *à* nor *de*, such as the agent in passive form, which is introduced by the preposition *par*.

We have added in the SF two elements that do not correspond to arguments of the verb: the reflexive pronoun, and the passive auxiliary. The reason for adding these elements to the SF is that their presence influences the presence or absence of some arguments of the verb, and therefore the SF.

The second important choice that must be made when defining SF is the *level of abstraction*, or, in other words, how much the SF abstracts away from its realization in the sentence. In our case, we have used two ways to abstract away from the surface realization of the SF. The first one is factoring several part of speech tags. We have factored pronouns, common nouns and proper nouns into a single category N. We have not gathered verbs in different modes into one category since the mode of the verb influences its syntactic behavior and hence its SF. The second means of abstraction we have used is the absence of linear order between the arguments. Taking into account argument order increases the number of SF and, hence, data sparseness, without adding much information for selecting the correct SF, this is why we have decided to to ignore it. In our second example above, each of the three arguments can be realized as one out of eight parts of speech that correspond to the part of speech tag N and the 24 possible orderings are represented as one canonical ordering. This SF therefore corresponds to $12\,288$ possible realizations.

3.2 Treebank Extracted Subcat Frames

This resource has been extracted from the TRAIN corpus. At a first glance, it may seen strange to extract data from the corpus that have been used for training our parser. The reason is that, as seen in section 1, SF are not directly modeled by the parser, which only takes into account subtrees made of, at most, two dependencies.

The extraction procedure of SF from the treebank is straightforward : the tree of every sentence is visited and, for every verb of the sentence, its daughters are visited, and, depending whether they are considered as arguments of the verb (with respect to the conventions or section 3.1), they are added to the SF. The number of different verbs extracted, as well as the number of different SF and the average number of SF per verb are displayed in table 3. Column T (for Train) is the one that we are interested in here.

	T	L	A_0	A_5	A_{10}
nb of verbs	2058	7824	23915	4871	3923
nb of diff SF	666	1469	12122	2064	1355
avg. nb of SF	4.83	52.09	14.26	16.16	13.45

Table 3: Resources statistics

The extracted resource can directly be compared with the TREELEX resource (Kupsc and Abeillé, 2008), which has been extracted from the same treebank. The result that we obtain is different, due to the fact that (Kupsc and Abeillé, 2008) have a more abstract definition of SF. As a consequence, they define a smaller number of SF: 58 instead of 666 in our case. The smaller number of SF yields a smaller average number of SF per verb: 1.72 instead of 4.83 in our case.

3.3 Automatically computed Subcat Frames

The extraction procedure described above has been used to extract ASF from an automatically parsed corpus. The corpus is actually a collection of three corpora of slightly different genres. The first one is a collection of news reports of the French press agency *Agence France Presse*, the second is a collection of newspaper articles from a local French newspaper : *l'Est Républicain*. The third one is a collection of articles from the French Wikipedia. The size of the different corpora are detailed in table 4.

The corpus was first POS tagged with the MELT tagger (Denis and Sagot, 2010), lemmatized with the MACAON tool suite (Nasr et al., 2011) and parsed in order to get the best parse for every sentence. Then the ASF have been extracted.

The number of verbs, number of SF and average number of SF per verb are represented in table 3, in column A_0 (A stands for Automatic). As one can see, the number of verbs and SF are unrealistic. This is due to the fact that the data that we extract SF from is noisy: it consists of automatically produced syntactic trees which contain errors (recall

CORPUS	Sent. nb.	Tokens nb.
AFP	2 041 146	59 914 238
EST REP	2 998 261	53 913 288
WIKI	1 592 035	33 821 460
TOTAL	5 198 642	147 648 986

Table 4: sizes of the corpora used to collect SF

that the LAS on the DEV corpus is $88,02\%$). There are two main sources of errors in the parsed data: the pre-processing chain (tokenization, part of speech tagging and lemmatization) which can consider as a verb a word that is not, and, of course, parsing errors, which tend to create crazy SF. In order to fight against noise, we have used a simple thresholding: we only collect ASF that occur more than a threshold i. The result of the thresholding appears in columns A_5 and A_{10} , where the subscript is the value of the threshold. As expected both the number of verbs and SF decrease sharply when increasing the value of the threshold.

Extracting SF for verbs from raw data has been an active direction of research for a long time, dating back at least to the work of (Brent, 1991) and (Manning, 1993). More recently (Messiant et al., 2008) proposed such a system for French verbs. The method we use for extracting SF is not novel with respect to such work. Our aim was not to devise new extraction techniques but merely to evaluate the resource produced by such techniques for statistical parsing.

3.4 Using an existing resource

The third resource that we have used is the Le*fff* (Lexique des formes fléchies du français — *Lexicon of French inflected form*), a large-coverage syntactic lexicon for French (Sagot, 2010). The Le*fff* was developed in a semi-automatic way: automatic tools were used together with manual work. The latest version of the Le*fff* contains 10,618 verbal entries for 7,835 distinct verbal lemmas (the Le*fff* covers all categories, but only verbal entries are used in this work).

A sub-categorization frame consists in a list of syntactic functions, using an inventory slightly more fine-grained than in the French Treebank, and for each of them a list of possible realizations (e.g., noun phrase, infinitive clause, or null-realization if the syntactic function is optional).

For each verbal lemma, we extracted all sub-categorization frames for each of the four verbal part-of-speech tags (V, VINF, VPR, VPP), thus creating an inventory of SFs in the same sense and format as described in Section 3.1. Note that such SFs do not contain alternatives concerning the way each syntactic argument is realized or not: this extraction process includes a de-factorization step. Its output, hereafter L, contains 801,246 distinct (lemma, SF) pairs.

3.5 Coverage

In order to be able to correct SF errors, the three resources described above must possess two important characteristics: high coverage and high accuracy. Coverage measures the presence, in the resource, of the correct SF of a verb, in a given sentence. Accuracy measures the ability of a resource to select the correct SF for a verb in a given context when several ones are possible.

We will give in this section coverage result, computed on the DEV corpus. Accuracy will be described and computed in section 4. The reason why the two measures are not described together is due to the fact that coverage can be computed on a reference corpus while accuracy must be computed on the output of a parser, since it is the parser that will propose different SF for a verb in a given context.

Given a reference corpus C and a resource R, two coverage measures have been computed, lexical coverage, which measures the ratio of verbs of C that appear in R and syntactic coverage, which measures the ratio of ASF of C that appear in R. Two variants of each measures are computed: on types and on occurrences. The values of these measures computed on the DEV corpus are summarized in table 5.

		T	L	A_0	A_5	A_{10}
Lex.	types	89.56	99.52	99.52	98.56	98.08
cov.	occ	96.98	99.85	99.85	99.62	99.50
Synt.	types	62.24	78.15	95.78	91.08	88.84
cov.	occ	73.54	80.35	97.13	93.96	92.39

Table 5: Lexical and syntactic coverage of the three resources on DEV

The figures of table 5 show that lexical coverage of the three resources is quite high, ranging

from 89.56 to 99.52 when computed on types and from 96.98 to 99.85 when computed on occurrences. The lowest coverage is obtained by the T resource, which does not come as a surprise since it is computed on a rather small number of sentences. It is also interesting to note that lexical coverage of A does not decrease much when augmenting the threshold, while the size of the resource decreases dramatically (as shown in table 3). This validates the hypothesis that the resource is very noisy and that a simple threshold on the occurrences of ASF is a reasonable means to fight against noise.

Syntactic coverage is, as expected, lower than lexical coverage. The best results are obtained by A_0: 95.78 on types and 97.13 on occurrences. Thresholding on the occurrences of anchored SF has a bigger impact on syntactic coverage than it had on lexical coverage. A threshold of 10 yields a coverage of 88.84 on types and 92.39 on occurrences.

4 Integrating Subcat Frames in the Parser

As already mentioned in section 1, SF usually exceed the domain of locality of the structures that are directly modeled by the parser. It is therefore difficult to integrate directly SF in the model of the parser. In order to circumvent the problem, we have decided to work on the n-best output of the parser: we consider that a verb v, in a given sentence S, can be associated to any of the SF that v licenses in one of the n-best trees. The main weakness of this method is that an SF error can be corrected only if the right SF appears at least in one of the n-best parse trees.

In order to estimate an upper bound of the SAS that such methods can reach (how many SF errors can actually be corrected), we have computed the oracle SAS on the 100 best trees of the DEV corpus DEV (for how many verbs the correct SF appears in at least one of the n-best parse trees). The oracle score is equal to 95.16, which means that for 95.16% of the verb occurrences of the DEV, the correct SF appears somewhere in the 100-best trees. 95.16 is therefore the best SAS that we can reach. Recall that the baseline SAS is equal to 79.88% the room for progress is therefore equal to 15.28% absolute.

Three experiments are described below. In the first one, section 4.1, a simple technique, called Post Processing is used. Section 4.2 describes a second technique, called Double Parsing, which is a is a refinement of Post Processing. Both sections 4.1 and 4.2 are based on single resources. Section 4.3 proposes a simple way to combine the different resources.

4.1 Post Processing

The post processing method (PP) is the simplest one that we have tested. It takes as input the different ASF that occur in the n-best output of the parser as well as a resource R. Given a sentence, let's note $T_1 \dots T_n$ the trees that appear in the n-best output of the parser, in decreasing order of their score. For every verb v of the sentence, we note $\mathcal{S}(v)$ the set of all the SF associated to v that appear in the trees $T_1 \dots T_n$.

Given a verb v and a SF s, we define the following functions:

$\mathcal{C}(v, s)$ is the number of occurrences of the ASF (v, s) in the trees $T_1 \dots T_n$.

$\mathcal{F}(v)$ is the SF associated to v in T_1

$\mathcal{C}_R(v, s)$ the number of occurrences of the ASF (v, s) in the resource R.

We define a selection function as a function that selects a SF for a given verb in a given sentence. A selection function has to take into account the information given by the resource (whether an SF is acceptable/frequent for a given verb) as well as the information given by the parser (whether the parser has a strong preference to associate a given SF to a given verb).

In our experiments, we have tested two simple selection functions. φ_R which selects the first SF $s \in \mathcal{S}(v)$, such that $\mathcal{C}_R(v, s) > 0$ when traversing the trees $T_1 \dots T_n$ in the decreasing order of score (best tree first).

The second function, $\psi_R(v)$ compares the most frequent SF for v in the resource R with the SF of the first parse. If the ratio of the number of occurrences in the n-best of the former and the latter is above a threshold α, the former is selected. More formally:

$$\psi_R(v) = \begin{cases} \hat{s} = \arg\max_{s \in \mathcal{S}(v)} \mathcal{C}_R(v, s) \\ \qquad \text{if } \frac{\mathcal{C}(v,\hat{s})}{\mathcal{C}(v,\mathcal{F}(v))} > \alpha \\ \mathcal{F}(v) \\ \qquad \text{otherwise} \end{cases}$$

The coefficient α has been optimized on DEV corpus. Its value is equal to 2.5 for the Automatic resource, 2 for the Train resource and 1.5 for the Lefff.

The construction of the new solution proceeds as follows: for every verb v of the sentence, a SF is selected with the selection function. It is important to note, at this point, that the SF selected for different verbs of the sentence can pertain to different parse trees. The new solution is built based on tree T_1. For every verb v, its arguments are potentially modified in agreement with the SF selected by the selection function. There is no guarantee at this point that the solution is well formed. We will return to this problem in section 4.2.

We have evaluated the PP method with different selection functions on the TEST corpus. The results of applying function ψ_R were more successful. As a result we just report the results of this function in table 6. Different levels of thresholding for resource A gave almost the same results, we therefore used A_{10} which is the smallest one.

	B	T	L	A
SAS	80.84	83.11	82.14	82.17
LAS	88.88	89.14	89.03	89.03
UAS	90.71	90.91	90.81	90.82

Table 6: LAS and UAS on TEST using PP

The results of table 6 show two interesting facts. First, the SAS is improved, it jumps from 80.84 to 83.11. PP therefore corrects some SF errors made by the parser. It must be noted however that this improvement is much lower than the oracle score. The second interesting fact is the very moderate increase of both LAS and UAS. This is due to the fact that the number of dependencies modified is small with respect to the total number of dependencies. The impact on LAS and UAS is therefore weak.

The best results are obtained with resource T. Although the coverage of T is low, the resource is very close to the train data, this fact probably explains the good results obtained with this resource.

It is interesting, at this point, to compare our method with a reranking approach. In order to do so, we have compared the upper bound of the number of SF errors that can be corrected when using reranking and our approach. The results of the comparison computed on a list of 100 best trees is reported in table 7 which shows the ratio of subcat frame errors that could be corrected with a reranking approach and the ratio of errors sub-parse recombining could reach.

	DEV	TEST
reranking	53.9%	58.5%
sub-parse recombining	75.5%	76%

Table 7: Correction rate for subcat frames errors with different methods

Table 7 shows that combining sub-parses can, in theory, correct a much larger number of wrong SF assignments than reranking.

4.2 Double Parsing

The post processing method shows some improvement over the baseline. But it has an important drawback: it can create inconsistent parses. Recall that the parser we are using is based on a second order model. In other words, the score of a dependency depends on some neighboring dependencies. When building a new solution, the post processing method modifies some dependencies independently of their context, which may give birth to very unlikely configurations.

In order to compute a new optimal parse tree that preserves the modified dependencies, we have used a technique proposed in (Mirroshandel and Nasr, 2011) that modifies the scoring function of the parser in such a way that the dependencies that we want to keep in the parser output get better scores than all competing dependencies. The new solution is therefore the optimal solution that preserves the dependencies modified by the PP method.

The double parsing (DP) method is therefore a three stage method. First, sentence S is parsed, producing the n-best parses. Then, the post processing method is used, modifying the first best parse. Let's note $\mathcal{D}$ the set of dependencies that were changed in this process. In the last stage, a new parse is produced, that preserves $\mathcal{D}$.

	B	T	L	A
SAS	80.84	83.11	82.14	82.17
LAS	88.88	89.30	89.25	89.31
UAS	90.71	91.07	91.05	91.08

Table 8: LAS and UAS on TEST using DP

The results of DP on TEST are reported in table 8. SAS did not change with respect to PP, because DP keeps the SF selected by PP. As expected DP does increase LAS and UAS. Recomputing an optimal solution therefore increases the quality of the parses. Table 8 also shows that the three resources get almost the same LAS and UAS although SAS is better for resource T.

4.3 Combining Resources

Due to the different generation techniques of our three resources, another direction of research is combining them. We did different experiments concerning all possible combination of resources: A and L (AL), T and L (TL), T and A (TA), and all tree (TAL) resources. The results of these combinations for PP and DP methods are shown in tables 9 and 10, respectively.

The resource are combined in a back-off schema: we search for a candidate ASF in a first resource. If it is found, the search stops. Otherwise, the next resource(s) are probed. One question that arises is: which sequence is the optimal one for combining the resources. To answer this question, we did several experiments on DEV set. Our experiments have shown that it is better to search T resource, then A, and, eventually, L. The results of this combining method, using PP are reported in table 9. The best results are obtained for the TL combination. The SAS jumps from 83.11 to 83.76. As it was the case with single resources, the LAS and UAS increase is moderate.

	B	AL	TL	TA	TAL
SAS	80.84	82.12	83.76	83.50	83.50
LAS	88.88	89.03	89.22	89.19	89.19
UAS	90.71	90.79	90.98	90.95	90.95

Table 9: LAS and UAS on TEST using PP with resource combination

With DP (table 9), the order of resource combination is exactly the same as with PP. As was the case with single resources, DP has a positive, but moderate, impact on LAS and UAS.

The results of tables 9 and 10 do not show considerable improvement over single resources. This might be due to the large intersection between our resources. In other words, they do not have complementary information, and their combination will not introduce much information. Another possible reason for this result is the combination technique used. More sophisticated techniques might yield better results.

	B	AL	TL	TA	TAL
SAS	80.84	82.12	83.76	83.50	83.50
LAS	88.88	89.22	89.31	89.34	89.34
UAS	90.71	91.02	91.05	91.08	91.09

Table 10: LAS and UAS on TEST using DP with resource combination

5 Conclusions

Subcategorization frames for verbs constitute a rich source of lexico-syntactic information which is hard to integrate in graph based parsers. In this paper, we have used three different resources for subcategorization frames. These resources are from different origins with various characteristics. We have proposed two different methods to introduce the useful information from these resources in a second order model parser. We have conducted different experiments on French Treebank that showed a 15.24% reduction of erroneous SF selections for verbs. Although encouraging, there is still plenty of room for better results since the oracle score for 100 best parses is equal to 95.16% SAS and we reached 83.76%. Future work will concentrate on more elaborate selection functions as well as more sophisticated ways to combine the different resources.

Acknowledgments

This work has been funded by the French Agence Nationale pour la Recherche, through the project EDYLEX (ANR-08-CORD-009).

References

A. Abeillé, L. Clément, and F. Toussenel. 2003. Building a treebank for french. In Anne Abeillé, editor, *Treebanks*. Kluwer, Dordrecht.

A. Arun and F. Keller. 2005. Lexicalization in crosslinguistic probabilistic parsing: The case of french. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 306–313. Association for Computational Linguistics.

B. Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of ACL*, pages 89–97.

Michael Brent. 1991. Automatic acquisition of subcategorization frames from untagged text. In *Proceedings of ACL*.

Joan Bresnan, editor. 1982. *The Mental Representation of Grammatical Relations*. MIT Press.

M. Candito, B. Crabbé, P. Denis, and F. Guérin. 2009. Analyse syntaxique du français : des constituants aux dépendances. In *Proceedings of Traitement Automatique des Langues Naturelles*.

J. Carroll, G. Minnen, and T. Briscoe. 1998. Can subcategorisation probabilities help a statistical parser? *Arxiv preprint cmp-lg/9806013*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of ACL*.

Michael Collins. 1997. Three Generative, Lexicalised Models for Statistical Parsing. In *Proceedings of the 35th Annual Meeting of the ACL*.

Michael Collins. 2000. Discriminative Reranking for Natural Language Parsing. In *Proceedings of ICML*.

P. Denis and B. Sagot. 2010. Exploitation d'une ressource lexicale pour la construction d'un étiqueteur morphosyntaxique état-de-l'art du français. In *Proceedings of Traitement Automatique des Langues Naturelles*.

Gerald Gazdar, Ewan Klein, Geoffrey K. Pullum, and Ivan Sag. 1985. *Generalized Phrase Structure Grammar*. Harvard University Press.

Aravind Joshi, Leon Levy, and M Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10:136–163.

Anna Kupsc and Anne Abeillé. 2008. Treelex: A subcategorisation lexicon for french verbs. In *Proceedings of the First International Conference on Global Interoperability for Language Resources*.

Christopher Manning. 1993. Automatic acquisition of a large subcategorization dictionary from corpora. In *Proceedings of ACL*.

M.P. Marcus, M.A. Marcinkiewicz, and B. Santorini. 1993. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 91–98.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP*, pages 523–530.

C. Messiant, A. Korhonen, T. Poibeau, et al. 2008. Lexschem: A large subcategorization lexicon for french verbs. In *Proceedings of the Language Resources and Evaluation Conference*.

S.A. Mirroshandel and A. Nasr. 2011. Active learning for dependency parsing using partially annotated sentences. In *Proceedings of International Conference on Parsing Technologies*.

A. Nasr, F. Béchet, J-F. Rey, B. Favre, and Le Roux J. 2011. MACAON: An NLP tool suite for processing word lattices. In *Proceedings of ACL*.

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

Slav Petrov and Dan Klein. 2008. Discriminative Log-Linear Grammars with Latent Variables. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, *Advances in Neural Information Processing Systems 20 (NIPS)*, pages 1153–1160, Cambridge, MA. MIT Press.

Carl Pollard and Ivan Sag. 1994. *Head-driven Phrase Structure Grammmar*. CSLI Series. University of Chicago Press.

Adwait Ratnaparkhi. 1999. Learning to parse natural language with maximum entropy models. *Machine learning*, 34(1):151–175.

Benoît Sagot. 2010. The Le*fff*, a freely available and large-coverage morphological and syntactic lexicon for french. In *Proceedings of the Seventh conference on International Language Resources and Evaluation (LREC'10)*, pages 2744–2751, Valletta, Malta.

D. Zeman. 2002. Can subcategorization help a statistical dependency parser? In *Proceedings of the 19th international conference on Computational linguistics-Volume 1*, pages 1–7. Association for Computational Linguistics.