Benchmarking of Statistical Dependency Parsers for French

Marie Candito⋆, Joakim Nivre◇, Pascal Denis⋆ and Enrique Henestroza Anguiano⋆
⋆ Alpage (Université Paris 7/INRIA)
◇ Uppsala University, Department of Linguistics and Philology
marie.candito@linguist.jussieu.fr {pascal.denis, henestro}@inria.fr joakim.nivre@lingfil.uu.se

Abstract

We compare the performance of three statistical parsing architectures on the problem of deriving typed dependency structures for French. The architectures are based on PCFGs with latent variables, graph-based dependency parsing and transition-based dependency parsing, respectively. We also study the influence of three types of lexical information: lemmas, morphological features, and word clusters. The results show that all three systems achieve competitive performance, with a best labeled attachment score over 88%. All three parsers benefit from the use of automatically derived lemmas, while morphological features seem to be less important. Word clusters have a positive effect primarily on the latent variable parser.

1 Introduction

In this paper, we compare three statistical parsers that produce typed dependencies for French. A syntactic analysis in terms of typed grammatical relations, whether encoded as functional annotations in syntagmatic trees or in labeled dependency trees, appears to be useful for many NLP tasks including question answering, information extraction, and lexical acquisition tasks like collocation extraction.

This usefulness holds particularly for French, a language for which bare syntagmatic trees are often syntactically underspecified because of a rather free order of post-verbal complements/adjuncts and the possibility of subject inversion. Thus, the annotation scheme of the French Treebank (Abeillé and Barrier, 2004) makes use of flat syntagmatic trees without VP nodes, with no structural distinction between complements, adjuncts or post-verbal subjects, but with additional functional annotations on dependents of verbs.

Parsing is commonly enhanced by using more abstract lexical information, in the form of morphological features (Tsarfaty, 2006), lemmas (Seddah et al., 2010), or various forms of clusters (see (Candito and Seddah, 2010) for references). In this paper, we explore the integration of morphological features, lemmas, and linear context clusters.

Typed dependencies can be derived using many different parsing architectures. As far as statistical approaches are concerned, the dominant paradigm for English has been to use constituency-based parsers, the output of which can be converted to typed dependencies using well-proven conversion procedures, as in the Stanford parser (Klein and Manning, 2003). In recent years, it has also become popular to use statistical dependency parsers, which are trained directly on labeled dependency trees and output such trees directly, such as MSTParser (McDonald, 2006) and MaltParser (Nivre et al., 2006). Dependency parsing has been applied to a fairly broad range of languages, especially in the CoNLL shared tasks in 2006 and 2007 (Buchholz and Marsi, 2006; Nivre et al., 2007).

We present a comparison of three statistical parsing architectures that output typed dependencies for French: one constituency-based architecture featuring the Berkeley parser (Petrov et al., 2006), and two dependency-based systems using radically different parsing methods, MSTParser (McDonald et al., 2006) and MaltParser (Nivre et al., 2006). These three systems are compared both in terms of parsing accuracy and parsing times, in realistic settings that only use predicted information. By using freely available software packages that implement language-independent approaches

and applying them to a language different from English, we also hope to shed some light on the capacity of different methods to cope with the challenges posed by different languages.

Comparative evaluation of constituency-based and dependency-based parsers with respect to labeled accuracy is rare, despite the fact that parser evaluation on typed dependencies has been advocated for a long time (Lin, 1995; Carroll et al., 1998). Early work on statistical dependency parsing often compared constituency-based and dependency-based methods with respect to their *unlabeled* accuracy (Yamada and Matsumoto, 2003), but comparison of different approaches with respect to *labeled* accuracy is more recent.

Cer et al. (2010) present a thorough analysis of the best trade-off between speed and accuracy in deriving Stanford typed dependencies for English (de Marneffe et al., 2006), comparing a number of constituency-based and dependency-based parsers on data from the Wall Street Journal. They conclude that the highest accuracy is obtained using constituency-based parsers, although some of the dependency-based parsers are more efficient.

For German, the 2008 ACL workshop on parsing German (Kübler, 2008) featured a shared task with two different tracks, one for constituency-based parsing and one for dependency-based parsing. Both tracks had their own evaluation metrics, but the accuracy with which parsers identified subjects, direct objects and indirect objects was compared across the two tracks, and the results in this case showed an advantage for dependency-based parsing.

In this paper, we contribute results for a third language, French, by benchmarking both constituency-based and dependency-based methods for deriving typed dependencies. In addition, we investigate the usefulness of morphological features, lemmas and word clusters for each of the different parsing architectures. The rest of the paper is structured as follows. Section 2 describes the French Treebank, and Section 3 describes the three parsing systems. Section 4 presents the experimental evaluation, and Section 5 contains a comparative error analysis of the three systems. Section 6 concludes with suggestions for future research.

2 Treebanks

For training and testing the statistical parsers, we use treebanks that are automatically converted from the French Treebank (Abeillé and Barrier, 2004) (hereafter FTB), a constituency-based treebank made up of $12,531$ sentences from the *Le Monde* newspaper. Each sentence is annotated with a constituent structure and words bear the following features: gender, number, mood, tense, person, definiteness, wh-feature, and clitic case. Nodes representing dependents of a verb are labeled with one of 8 grammatical functions.[1]

We use two treebanks automatically obtained from FTB, both described in Candito et al. (2010). FTB-UC is a modified version of the original constituency-based treebank, where the rich morphological annotation has been mapped to a simple tagset of 28 part-of-speech tags, and where compounds with regular syntax are broken down into phrases containing several simple words while remaining sequences annotated as compounds in FTB are merged into a single token. Function labels are appended to syntactic category symbols and are either used or ignored, depending on the task.

FTB-UC-DEP is a dependency treebank derived from FTB-UC using the classic technique of head propagation rules, first proposed for English by Magerman (1995). Function labels that are present in the original treebank serve to label the corresponding dependencies. The remaining unlabeled dependencies are labeled using heuristics (for dependents of non-verbal heads). With this conversion technique, output dependency trees are necessarily projective, and extracted dependencies are necessarily local to a phrase, which means that the automatically converted trees can be regarded as pseudo-projective approximations to the correct dependency trees (Kahane et al., 1998). Candito et al. (2010) evaluated the converted trees for 120 sentences, and report a 98% labeled attachment score when comparing the automatically converted dependency trees to the manually corrected ones.

[1] These are `SUJ` (subject), `OBJ` (object), `A-OBJ`/`DE-OBJ` (indirect object with preposition *à* / *de*), `P-OBJ` (indirect object with another preposition / locatives), `MOD` (modifier), `ATS`/`ATO` (subject/object predicative complement).

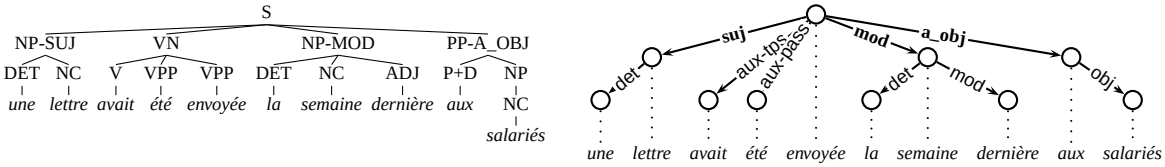

Figure 1: An example of constituency tree of the FTB-UC (left), and the corresponding dependency tree (right) for *A letter had been sent the week before to the employees*.

Figure 1 shows two parallel trees from FTB-UC and FTB-UC-DEP. In all reported experiments in this paper, we use the usual split of FTB-UC: first 10% as test set, next 10% as dev set, and the remaining sentences as training set.

3 Parsers

Although all three parsers compared are statistical, they are based on fairly different parsing methodologies. The Berkeley parser (Petrov et al., 2006) is a latent-variable PCFG parser, MSTParser (McDonald et al., 2006) is a graph-based dependency parser, and MaltParser (Nivre et al., 2006) is a transition-based dependency parser.

The choice to include two different dependency parsers but only one constituency-based parser is motivated by the study of Seddah et al. (2009), where a number of constituency-based statistical parsers were evaluated on French, including Dan Bikel's implementation of the Collins parser (Bikel, 2002) and the Charniak parser (Charniak, 2000). The evaluation showed that the Berkeley parser had significantly better performance for French than the other parsers, whether measured using a parseval-style labeled bracketing F-score or a CoNLL-style unlabeled attachment score. Contrary to most of the other parsers in that study, the Berkeley parser has the advantage of a strict separation of parsing model and linguistic constraints: linguistic information is encoded in the treebank only, except for a language-dependent suffix list used for handling unknown words.

In this study, we compare the Berkeley parser to MSTParser and MaltParser, which have the same separation of parsing model and linguistic representation, but which are trained directly on labeled dependency trees. The two dependency parsers use radically different parsing approaches but have achieved very similar performance for a wide range of languages (McDonald and Nivre, 2007). We describe below the three architectures in more detail.[2]

3.1 The Berkeley Parser

The Berkeley parser is a freely available implementation of the statistical training and parsing algorithms described in (Petrov et al., 2006) and (Petrov and Klein, 2007). It exploits the fact that PCFG learning can be improved by splitting symbols according to structural and/or lexical properties (Klein and Manning, 2003). Following Matsuzaki et al. (2005), the Berkeley learning algorithm uses EM to estimate probabilities on symbols that are automatically augmented with latent annotations, a process that can be viewed as symbol splitting. Petrov et al. (2006) proposed to score the splits in order to retain only the most beneficial ones, and keep the grammar size manageable: the splits that induce the smallest losses in the likelihood of the treebank are merged back. The algorithm starts with a very general treebank-induced binarized PCFG, with order h horizontal markovisation. created, where at each level a symbol appears without track of its original siblings. Then the Berkeley algorithm performs split/merge/smooth cycles that iteratively refine the binarized grammar: it adds two latent annotations on each symbol, learns probabilities for the refined grammar, merges back 50% of the splits, and smoothes the final probabilities to prevent overfitting. All our experiments are run using BerkeleyParser 1.0,[3] modified for handling

[2] For replicability, models, preprocessing tools and experimental settings are available at `http://alpage.inria.fr/statgram/frdep.html`.

[3] `http://www.eecs.berkeley.edu/\~petrov/berkeleyParser`

French unknown words by Crabbé and Candito (2008), with otherwise default settings (order 0 horizontal markovisation, order 1 vertical markovisation, 5 split/merge cycles).

The Berkeley parser could in principle be trained on functionally annotated phrase-structure trees (as shown in the left half of figure 1), but Crabbé and Candito (2008) have shown that this leads to very low performance, because the splitting of symbols according to grammatical functions renders the data too sparse. Therefore, the Berkeley parser was trained on FTB-UC without functional annotation. Labeled dependency trees were then derived from the phrase-structure trees output by the parser in two steps: *(1)* function labels are assigned to phrase structure nodes that have functional annotation in the FTB scheme; and *(2)* dependency trees are produced using the same procedure used to produce the pseudo-gold dependency treebank from the FTB (cf. Section 2).

The functional labeling relies on the Maximum Entropy labeler described in Candito et al. (2010), which encodes the problem of functional labeling as a multiclass classification problem. Specifically, each class is of the eight grammatical functions used in FTB, and each head-dependent pair is treated as an independent event. The feature set used in the labeler attempt to capture bilexical dependencies between the head and the dependent (using stemmed word forms, parts of speech, etc.) as well as more global sentence properties like mood, voice and inversion.

3.2 MSTParser

MSTParser is a freely available implementation of the parsing models described in McDonald (2006). These models are often described as *graph-based* because they reduce the problem of parsing a sentence to the problem of finding a directed maximum spanning tree in a dense graph representation of the sentence. Graph-based parsers typically use global training algorithms, where the goal is to learn to score correct trees higher than incorrect trees. At parsing time a global search is run to find the highest scoring dependency tree. However, unrestricted global inference for graph-based dependency parsing is NP-hard, and graph-based parsers like MSTParser therefore limit the scope of their features to a small number of adjacent arcs (usually two) and/or resort to approximate inference (McDonald and Pereira, 2006). For our experiments, we use MSTParser 0.4.3b[4] with 1-best projective decoding, using the algorithm of Eisner (1996), and second order features. The labeling of dependencies is performed as a separate sequence classification step, following McDonald et al. (2006).

To provide part-of-speech tags to MSTParser, we use the MElt tagger (Denis and Sagot, 2009), a Maximum Entropy Markov Model tagger enriched with information from a large-scale dictionary.[5] The tagger was trained on the training set to provide POS tags for the dev and test sets, and we used 10-way jackknifing to generate tags for the training set.

3.3 MaltParser

MaltParser[6] is a freely available implementation of the parsing models described in (Nivre, 2006) and (Nivre, 2008). These models are often characterized as *transition-based*, because they reduce the problem of parsing a sentence to the problem of finding an optimal path through an abstract transition system, or state machine. This is sometimes equated with shift-reduce parsing, but in fact includes a much broader range of transition systems (Nivre, 2008). Transition-based parsers learn models that predict the next state given the current state of the system, including features over the history of parsing decisions and the input sentence. At parsing time, the parser starts in an initial state and greedily moves to subsequent states – based on the predictions of the model – until a terminal state is reached. The greedy, deterministic parsing strategy results in highly efficient parsing, with run-times often linear in sentence length, and also facilitates the use of arbitrary non-local features, since the partially built dependency tree is fixed in any given state. However, greedy inference can also lead to error propagation if early predictions place the parser in incorrect states. For the experiments in this paper, we use MaltParser

[4] `http://mstparser.sourceforge.net`

[5] Denis and Sagot (2009) report a tagging accuracy of 97.7% (90.1% on unknown words) on the FTB-UC test set.

[6] `http://www.maltparser.org`

1.3.1 with the arc-eager algorithm (Nivre, 2008) and use linear classifiers from the LIBLINEAR package (Fan et al., 2008) to predict the next state transitions. As for MST, we used the MElt tagger to provide input part-of-speech tags to the parser.

4 Experiments

This section presents the parsing experiments that were carried out in order to assess the state of the art in labeled dependency parsing for French and at the same time investigate the impact of different types of lexical information on parsing accuracy. We present the features given to the parsers, discuss how they were extracted/computed and integrated within each parsing architecture, and then summarize the performance scores for the different parsers and feature configurations.

4.1 Experimental Space

Our experiments focus on three types of lexical features that are used either in addition to or as substitutes for word forms: morphological features, lemmas, and word clusters. In the case of MaltParser and MSTParser, these features are used in conjunction with POS tags. Motivations for these features are rooted in the fact that French has a rather rich inflectional morphology.

The intuition behind using morphological features like tense, mood, gender, number, and person is that some of these are likely to provide *additional cues* for syntactic attachment or function type. This is especially true given that the 29 tags used by the MElt tagger are rather coarse-grained.

The use of lemmas and word clusters, on the other hand, is motivated by *data sparseness* considerations: these provide various degrees of generalization over word forms. As suggested by Koo et al. (2008), the use of word clusters may also reduce the need for annotated data.

All our features are automatically produced: no features except word forms originate from the treebank. Our aim was to assess the performance currently available for French in a realistic setting.

Lemmas Lemmatized forms are extracted using Le*fff* (Sagot, 2010), a large-coverage morphosyntactic lexicon for French, and a set of heuristics for unknown words. More specifically, Le*fff* is queried for each $(word, pos)$, where pos is the tag predicted by the MElt tagger. If the pair is found, we use the longest lemma associated with it in Le*fff*. Otherwise, we rely on a set of simple stemming heuristics using the form and the predicted tag to produce the lemma. We use the form itself for all other remaining cases.[7]

Morphological Features Morphological features were extracted in a way similar to lemmas, again by querying Le*fff* and relying on heuristics for out-of-dictionary words. Here are the main morphological attributes that were extracted from the lexicon: mood and tense for verbs; person for verbs and pronouns; number and gender for nouns, past participles, adjectives and pronouns; whether an adverb is negative; whether an adjective, pronoun or determiner is cardinal, ordinal, definite, possessive or relative. Our goal was to predict all attributes found in FTB that are recoverable from the word form alone.

Word Form Clusters Koo et al. (2008) have proposed to use unsupervised word clusters as features in MSTParser, for parsing English and Czech. Candito and Crabbé (2009) showed that, for parsing French with the Berkeley parser, using the same kind of clusters as substitutes for word forms improves performance. We now extend their work by comparing the impact of such clusters on two additional parsers.

We use the word clusters computed by Candito and Crabbé (2009) using Percy Liang's implementation[8] of the Brown unsupervised clustering algorithm (Brown et al., 1992). It is a bottom-up hierarchical clustering algorithm that uses a bigram language model over clusters. The resulting cluster ids are bit-strings, and various levels of granularity can be obtained by retaining only the first x bits. Candito and Crabbé (2009) used the *L'Est Républicain* corpus, a 125 million word journalistic corpus.[9] To reduce lexi-

[7] Candito and Seddah (2010) report the following coverage for the Le*fff*: around 96% of the tokens, and 80.1% of the token types are present in the Le*fff* (leaving out punctuation and numeric tokens, and ignoring case differences).

[8] `http://www.eecs.berkeley.edu/~pliang/software`

[9] `http://www.cnrtl.fr/corpus/estrepublicain`

cal data sparseness caused by inflection, they ran a lexicon-based stemming process on the corpus that removes inflection marks without adding or removing lexical ambiguity. The Brown algorithm was then used to compute 1000 clusters of stemmed forms, limited to forms that appeared at least 20 times.

We tested the use of clusters with different values for two parameters: **nbbits** = the cluster prefix length in bits, to test varying granularities, and **minocc** = the minimum number of occurrences in the *L'Est Républicain* corpus for a form to be replaced by a cluster or for a cluster feature to be used for that form.

4.2 Parser-Specific Configurations

Since the three parsers are based on different machine learning algorithms and parsing algorithms (with different memory requirements and parsing times), we cannot integrate the different features described above in exactly the same way. For the Berkeley parser we use the setup of Candito and Seddah (2010), where additional information is encoded within symbols that are used as substitutes for word forms. For MaltParser and MSTParser, which are based on discriminative models that permit the inclusion of interdependent features, additional information may be used either in addition to or as substitutes for word forms. Below we summarize the configurations that have been explored for each parser:

- **Berkeley:**
  1. **Morphological features:** N/A.
  2. **Lemmas:** Concatenated with POS tags and substituted for word forms.
  3. **Clusters:** Concatenated with morphological suffixes and substituted for word forms; grid search for optimal values of **nbbits** and **minocc**.
- **MaltParser** and **MSTParser:**
  1. **Morphological features:** Added as features.
  2. **Lemmas:** Substituted for word forms or added as features.
  3. **Clusters:** Substituted for word forms or added as features; grid search for optimal values of **nbbits** and **minocc**.

4.3 Results

Table 1 summarizes the experimental results. For each parser we give results on the development set for the baseline (no additional features), the best configuration for each individual feature type, and the best configuration for any allowed combination of the three features types. For the final test set, we only evaluate the baseline and the best combination of features. Scores on the test set were compared using a χ^2-test to assess statistical significance: unless specified, all differences therein were significant at $p \leq 0.01$.

The MSTParser system achieves the best labeled accuracy on both the development set and the test set. When adding lemmas, the best configuration is to use them as substitutes for word forms, which slightly improves the UAS results. For the clusters, their use as substitutes for word forms tends to degrade results, whereas using them as features alone has almost no impact. This means that we could not replicate the positive effect[10] reported by Koo et al. (2008) for English and Czech. However, the best combined configuration is obtained using lemmas instead of words, a reduced set of morphological features,[11] and clusters as features, with minocc=$50,000$ and nbbits=10.

MaltParser has the second best labeled accuracy on both the development set and the test set, although the difference with Berkeley is not significant on the latter. MaltParser has the lowest unlabeled accuracy of all three parsers on both datasets. As opposed to MSTParser, all three feature types work best for MaltParser when used in addition to word forms, although the improvement is statistically significant only for lemmas and clusters. Again, the best model uses all three types of features, with cluster features minocc=600 and nbbits=7. MaltParser shows the smallest discrepancy from unlabeled to labeled scores. This might be because it is the only architecture where labeling is directly done as part of parsing.

[10] Note that the two experiments cannot be directly compared. Koo et al. (2008) use their own implementation of an MST parser, which includes extra second-order features (e.g. grand-parent features on top of sibling features).

[11] As MSTParser training is memory-intensive, we removed the features containing information already encoded part-of-speech tags.

	Development Set										Test Set			
	Baseline		**Morpho**		**Lemma**		**Cluster**		**Best**		**Baseline**		**Best**	
Parser	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS	UAS
Berkeley	85.1	89.3	–	–	85.9	90.0	86.5	90.8	86.5	**90.8**	85.6	89.6	86.8	**91.0**
MSTParser	**87.2**	**90.0**	87.2	90.2	87.2	90.1	87.2	90.1	**87.5**	90.3	87.6	90.3	**88.2**	90.9
MaltParser	86.2	89.0	86.3	89.0	86.6	89.2	86.5	89.2	86.9	89.4	86.7	89.3	87.3	89.7

Table 1: Experimental results for the three parsing systems. LAS=labeled accuracy, UAS=unlabeled accuracy, for sentences of any length, ignoring punctuation tokens. Morpho/Lemma/Cluster=best configuration when using morphological features only (resp. lemmas only, clusters only), Best=best configuration using any combination of these.

For Berkeley, the lemmas improve the results over the baseline, and its performance reaches that of MSTParser for unlabeled accuracy (although the difference between the two parsers is not significant on the test set). The best setting is obtained with clusters instead of word forms, using the full bit strings. It also gives the best unlabeled accuracy of all three systems on both the development set and the test set. For the more important labeled accuracy, the point-wise labeler used is not effective enough.

Overall, MSTParser has the highest labeled accuracy and Berkeley the highest unlabeled accuracy. However, results for all three systems on the test set are roughly within one percentage point for both labeled and unlabeled accuracy, which means that we do not find the same discrepancy between constituency-based and dependency-based parser that was reported for English by Cer et al. (2010).

Table 2 gives parsing times for the best configuration of each parsing architecture. MaltParser runs approximately 9 times faster than the Berkeley system, and 10 times faster than MSTParser. The difference in efficiency is mainly due to the fact that MaltParser uses a linear-time parsing algorithm, while the other two parsers have cubic time complexity. Given the rather small difference in labeled accuracy, MaltParser seems to be a good choice for processing very large corpora.

	Bky	**Malt**	**MST**
Tagging	_	0:27	0:27
Parsing	12:19	0:58 (0:18)	14:12 (12:44)
Func. Lab.	0:23	_	_
Dep. Conv.	0:4	_	_
Total	12:46	1:25	14:39

Table 2: Parsing times (min:sec) for the dev set, for the three architectures, on an imac 2.66GHz. The figures within brackets show the pure parsing time without the model loading time, when available.

5 Error Analysis

We provide a brief analysis of the errors made by the best performing models for Berkeley, MSTParser and MaltParser on the development set, focusing on labeled and unlabeled attachment for nouns, prepositions and verbs. For nouns, Berkeley has the best unlabeled attachment, followed by MSTParser and then MaltParser, while for labeled attachment Berkeley and MSTParser are on a par with MaltParser a bit behind. For prepositions, MSTParser is by far the best for both labeled and unlabeled attachment, with Berkeley and MaltParser performing equally well on unlabeled attachment and MaltParser performing better than Berkeley on labeled attachment.[12] For verbs, Berkeley has the best performance on both labeled and unlabeled attachment, with MSTParser and MaltParser performing about equally well. Although Berkeley has the best unlabeled attachment overall, it also has the worst labeled attachment, and we found that this is largely due to the functional role labeler having trouble assigning the correct label when the dependent is a preposition or a clitic.

For errors in attachment as a function of word distance, we find that precision and recall on dependencies of length > 2 tend to degrade faster for MaltParser than for MSTParser and Berkeley,

[12] In the dev set, for MSTParser, 29% of the tokens that do not receive the correct governor are prepositions (883 out of 3051 errors), while these represent 34% for Berkeley (992 out of 2914), and 30% for MaltParser (1016 out of 3340).

with Berkeley being the most robust for dependencies of length > 6. In addition, Berkeley is best at finding the correct root of sentences, while MaltParser often predicts more than one root for a given sentence. The behavior of MSTParser and MaltParser in this respect is consistent with the results of McDonald and Nivre (2007).

6 Conclusion

We have evaluated three statistical parsing architectures for deriving typed dependencies for French. The best result obtained is a labeled attachment score of 88.2%, which is roughly on a par with the best performance reported by Cer et al. (2010) for parsing English to Stanford dependencies. Note two important differences between their results and ours: First, the Stanford dependencies are in a way deeper than the surface dependencies tested in our work. Secondly, we find that for French there is no consistent trend favoring either constituency-based or dependency-based methods, since they achieve comparable results both for labeled and unlabeled dependencies.

Indeed, the differences between parsing architectures are generally small. The best performance is achieved using MSTParser, enhanced with predicted part-of-speech tags, lemmas, morphological features, and unsupervised clusters of word forms. MaltParser achieves slightly lower labeled accuracy, but is probably the best option if speed is crucial. The Berkeley parser has high accuracy for unlabeled dependencies, but the current labeling method does not achieve a comparably high labeled accuracy.

Examining the use of lexical features, we find that predicted lemmas are useful in all three architectures, while morphological features have a marginal effect on the two dependency parsers (they are not used by the Berkeley parser). Unsupervised word clusters, finally, give a significant improvement for the Berkeley parser, but have a rather small effect for the dependency parsers.

Other results for statistical dependency parsing of French include the pilot study of Candito et al. (2010), and the work ofSchluter and van Genabith (2009), which resulted in an LFG statistical French parser. However, the latter's results are obtained on a modified subset of the FTB, and are expressed in terms of F-score on LFG f-structure features, which are not comparable to our attachment scores. There also exist a number of grammar-based parsers, evaluated on gold test sets annotated with chunks and dependencies (Paroubek et al., 2005; de la Clergerie et al., 2008). Their annotation scheme is different from that of the FTB, but we plan to evaluate the statistical parsers on the same data in order to compare the performance of grammar-based and statistical approaches.

Acknowledgments

The first, third and fourth authors' work was supported by ANR Sequoia (ANR-08-EMER-013). We are grateful to our anonymous reviewers for their comments.

References

Abeillé, A. and N. Barrier. 2004. Enriching a french treebank. In *LREC'04*.

Bikel, D. M. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *HLT-02*.

Brown, P., V. Della Pietra, P. Desouza, J. Lai, and R. Mercer. 1992. Class-based n-gram models of natural language. *Computational linguistics*, 18(4).

Buchholz, S. and E. Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *CoNLL 2006*.

Candito, M. and B. Crabbé. 2009. Improving generative statistical parsing with semi-supervised word clustering. In *IWPT'09*.

Candito, M. and D. Seddah. 2010. Parsing word clusters. In *NAACL/HLT Workshop SPMRL 2010*.

Candito, M., B. Crabbé, and P. Denis. 2010. Statistical french dependency parsing : treebank conversion and first results. In *LREC 2010*.

Carroll, J., E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: A survey and a new proposal. In *LREC 1998*.

Cer, D., M.-C. de Marneffe, D. Jurafsky, and C. Manning. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *LREC 2010*.

Charniak, E. 2000. A maximum entropy inspired parser. In *NAACL 2000*.

Crabbé, B. and M. Candito. 2008. Expériences d'analyse syntaxique statistique du français. In *TALN 2008*.

de la Clergerie, E. V., C. Ayache, G. de Chalendar, G. Francopoulo, C. Gardent, and P. Paroubek. 2008. Large scale production of syntactic annotations for french. In *First International Workshop on Automated Syntactic Annotations for Interoperable Language Resources*.

de Marneffe, M.-C., B. MacCartney, and C. D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

Denis, P. and B. Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art pos tagging with less human effort. In *PACLIC 2009*.

Eisner, J. 1996. Three new probabilistic models for dependency parsing: An exploration. In *COLING 1996*.

Fan, R.-E., K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. 2008. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9.

Kahane, S., A. Nasr, and O. Rambow. 1998. Pseudo-projectivity: A polynomially parsable non-projective dependency grammar. In *ACL/COLING 1998*.

Klein, D. and C. D. Manning. 2003. Accurate unlexicalized parsing. In *ACL 2003*.

Koo, T., X. Carreras, and M. Collins. 2008. Simple semi-supervised dependency parsing. In *ACL-08:HLT*.

Kübler, S. 2008. The PaGe 2008 shared task on parsing german. In *ACL-08 Workshop on Parsing German*.

Lin, D. 1995. A dependency-based method for evaluating broad-coverage parsers. In *IJCAI-95*.

Magerman, D. M. 1995. Statistical decision-tree models for parsing. In *ACL 1995*.

Matsuzaki, T., Y. Miyao, and J. Tsujii. 2005. Probabilistic cfg with latent annotations. In *ACL 2005*.

McDonald, R. and J. Nivre. 2007. Characterizing the errors of data-driven dependency parsing models. In *EMNLP-CoNLL 2007*.

McDonald, R. and F. Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *EACL 2006*.

McDonald, R., K. Lerman, and F. Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *CoNLL 2006*.

McDonald, R. 2006. *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Nivre, J., Johan Hall, and Jens Nilsson. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *LREC 2006*.

Nivre, J., J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007. The CoNLL 2007 shared task on dependency parsing. In *CoNLL Shared Task of EMNLP-CoNLL 2007*.

Nivre, J. 2006. *Inductive Dependency Parsing*. Springer.

Nivre, J. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34.

Paroubek, P., L.-G. Pouillot, I. Robba, and A. Vilnat. 2005. Easy : Campagne d'évaluation des analyseurs syntaxiques. In *TALN 2005, EASy workshop : campagne d'évaluation des analyseurs syntaxiques*.

Petrov, S. and D. Klein. 2007. Improved inference for unlexicalized parsing. In *NAACL-07: HLT*.

Petrov, S., L. Barrett, R. Thibaux, and D. Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL 2006*.

Sagot, B. 2010. The Le*fff*, a freely available and large-coverage morphological and syntactic lexicon for french. In *LREC 2010*.

Schluter, N. and J. van Genabith. 2009. Dependency parsing resources for french: Converting acquired lfg f-structure. In *NODALIDA 2009*.

Seddah, D., M. Candito, and B. Crabbé. 2009. Cross parser evaluation and tagset variation: a french treebank study. In *IWPT 2009*.

Seddah, D., G. Chrupała, O. Cetinoglu, J. van Genabith, and M. Candito. 2010. Lemmatization and statistical lexicalized parsing of morphologically-rich languages. In *NAACL/HLT Workshop SPMRL 2010*.

Tsarfaty, R. 2006. Integrated morphological and syntactic disambiguation for modern hebrew. In *COLING/ACL 2006 Student Research Workshop*.

Yamada, H. and Y. Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *IWPT 2003*.