Generative Constituent Parsing and Discriminative Dependency Reranking: Experiments on English and French

Joseph Le Roux[◇] **Benoît Favre**[†] **Alexis Nasr**[†] **Seyed Abolghasem Mirroshandel**[†,⋆]
[◇]LIPN, Université Paris Nord – CNRS UMR 7030, Villetaneuse, France
[†]LIF, Université Aix-Marseille – CNRS UMR 7279, Marseille, France
[⋆]Computer Engineering Department, Sharif university of Technology, Tehran, Iran
leroux@univ-paris13.fr, benoit.favre@lif.univ-mrs.fr,
alexis.nasr@lif.univ-mrs.fr, ghasem.mirroshandel@lif.univ-mrs.fr

Abstract

We present an architecture for parsing in two steps. A phrase-structure parser builds for each sentence an n-best list of analyses which are converted to dependency trees. These dependency structures are then rescored by a discriminative reranker. Our method is language agnostic and enables the incorporation of additional information which are useful for the choice of the best parse candidate. We test our approach on the the Penn Treebank and the French Treebank. Evaluation shows a significative improvement on different parse metrics.

1 Introduction

Two competing approaches exist for parsing natural language. The first one, called generative, is based on the theory of formal languages and rewriting systems. Parsing is defined here as a process that transforms a string into a tree or a tree forest. It is often grounded on phrase-based grammars – although there are generative dependency parsers – in particular context-free grammars or one of their numerous variants, that can be parsed in polynomial time. However, the independence hypothesis that underlies this kind of formal system does not allow for precise analyses of some linguistic phenomena, such as long distance and lexical dependencies.

In the second approach, known as discriminative, the grammar is viewed as a system of constraints over the correct syntactic structures, the words of the sentence themselves being seen as constraints over the position they occupy in the sentence. Parsing boils down to finding a solution that is compatible with the different constraints. The major problem of this approach lies in its complexity. The constraints can, theoretically, range over any aspect of the final structures, which prevents from using efficient dynamic programming techniques when searching for a global solution. In the worst case, final structures must be enumerated in order to be evaluated. Therefore, only a subset of constraints is used in implementations for complexity reasons. This approach can itself be divided into formalisms relying on logic to describe constraints, as the model-theoretic syntax (Pullum and Scholz, 2001), or numerical formalisms that associate weights to lexico-syntactic substructures. The latter has been the object of some recent work thanks to progresses achieved in the field of Machine Learning. A parse tree is represented as a vector of features and its accuracy is measured as the distance between this vector and the reference.

One way to take advantage of both approaches is to combine them sequentially, as initially proposed by Collins (2000). A generative parser produces a set of candidates structures that constitute the input of a second, discriminative module, whose search space is limited to this set of candidates. Such an approach, parsing followed by reranking, is used in the Brown parser (Charniak and Johnson, 2005). The approach can be extended in order to feed the reranker with the output of different parsers, as shown by (Johnson and Ural, 2010; Zhang et al., 2009).

In this paper we are interested in applying reranking to dependency structures. The main reason is that many linguistic constraints are straightforward to implement on dependency structures, as, for example, subcategorization frames or selectional constraints that are closely linked to the notion of de-

pendents of a predicate. On the other hand, dependencies extracted from constituent parses are known to be more accurate than dependencies obtained from dependency parsers. Therefore the solution we choose is an indirect one: we use a phrase-based parser to generate n-best lists and convert them to lists of dependency structures that are reranked. This approach can be seen as trade-off between phrase-based reranking experiments (Collins, 2000) and the approach of Carreras et al. (2008) where a discriminative model is used to score lexical features representing unlabelled dependencies in the Tree Adjoining Grammar formalism.

Our architecture, illustrated in Figure 1, is based on two steps. During the first step, a syntagmatic parser processes the input sentence and produces n-best parses as well as their probabilities. They are annotated with a functional tagger which tags syntagms with standard syntactic functions *subject, object, indirect object* ... and converted to dependency structures by application of percolation rules. In the second step, we extract a set of features from the dependency parses and the associated probabilities. These features are used to reorder the n-best list and select a potentially more accurate parse. Syntagmatic parses are produced by the implementation of a PCFG-LA parser of (Attia et al., 2010), similar to (Petrov et al., 2006), a functional tagger and dependency converter for the target language. The reranking model is a linear model trained with an implementation of the MIRA algorithm (Crammer et al., 2006)[1].

Charniak and Johnson (2005) and Collins (2000) rerank phrase-structure parses and they also include head-dependent information, in other words unlabelled dependencies. In our approach we take into account grammatical functions or labelled dependencies.

It should be noted that the features we use are very generic and do not depend on the linguistic knowledge of the authors. We applied our method to English, the de facto standard for testing parsing technologies, and French which exhibits many aspects of a morphologically rich language. But our approach could be applied to other languages, provided that the resources – treebanks and conversion tools – exist.

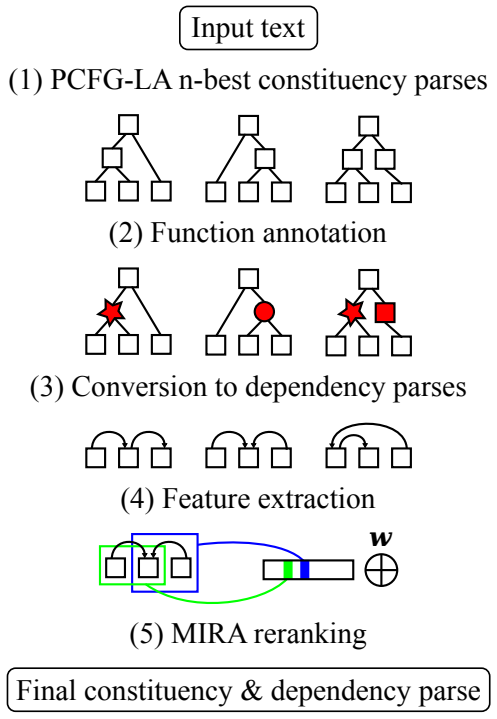


Figure 1: The parsing architecture: production of the n-best syntagmatic trees (1) tagged with functional labels (2), conversion to a dependency structure (3) and feature extraction (4), scoring with a linear model (5). The parse with the best score is considered as final.

The structure of the paper is the following: in Section 2 we describe the details of our generative parser and in Section 3 our reranking model together with the features templates. Section 4 reports the results of the experiments conducted on the Penn Treebank (Marcus et al., 1994) as well as on the Paris 7 Treebank (Abeillé et al., 2003) and Section 5 concludes the paper.

2 Generative Model

The first part of our system, the syntactic analysis itself, generates surface dependency structures in a sequential fashion (Candito et al., 2010b; Candito et al., 2010a). A phrase structure parser based on Latent Variable PCFGs (PCFG-LAs) produces tree structures that are enriched with functions and then converted to labelled dependency structures, which will be processed by the parse reranker.

[1]This implementation is available at `https://github.com/jihelhere/adMIRAble`.

2.1 PCFG-LAS

Probabilistic Context Free Grammars with Latent Annotations, introduced in (Matsuzaki et al., 2005) can be seen as automatically specialised PCFGs learnt from treebanks. Each symbol of the grammar is enriched with annotation symbols behaving as subclasses of this symbol. More formally, the probability of an unannotated tree is the sum of the probabilities of its annotated counterparts. For a PCFG-LA G, R is the set of annotated rules, $D(t)$ is the set of (annotated) derivations of an unannotated tree t, and $R(d)$ is the set of rules used in a derivation d. Then the probability assigned by G to t is:

$$P_G(t) = \sum_{d \in D(t)} P_G(d) = \sum_{d \in D(t)} \prod_{r \in R(d)} P_G(r) \quad (1)$$

Because of this alternation of sums and products that cannot be optimally factorised, there is no exact polynomial dynamic programming algorithm for parsing. Matsuzaki et al. (2005) and Petrov and Klein (2007) discuss approximations of the decoding step based on a Bayesian variational approach. This enables cubic time decoding that can be further enhanced with coarse-to-fine methods (Charniak and Johnson, 2005).

This type of grammars has already been tested on a variety of languages, in particular English and French, giving state-of-the-art results. Let us stress that this phrase-structure formalism is not lexicalised as opposed to grammars previously used in reranking experiments (Collins, 2000; Charniak and Johnson, 2005). The notion of lexical head is therefore absent at parsing time and will become available only at the reranking step.

2.2 Dependency Structures

A syntactic theory can either be expressed with phrase structures or dependencies, as advocated for in (Rambow, 2010). However, some information may be simpler to describe in one of the representations. This equivalence between the modes of representations only stands if the informational contents are the same. Unfortunately, this is not the case here because the phrase structures that we use do not contain functional annotations and lexical heads, whereas labelled dependencies do.

This implies that, in order to be converted into labelled dependency structures, phrase structure parses must first be annotated with functions. Previous experiments for English and French as well (Candito et al., 2010b) showed that a sequential approach is better than an integrated one for context-free grammars, because the strong independence hypothesis of this formalism implies a restricted domain of locality which cannot express the context needed to properly assign functions. Most functional taggers, such as the ones used in the following experiments, rely on classifiers whose feature sets can describe the whole context of a node in order to make a decision.

3 Discriminative model

Our discriminative model is a linear model trained with the Margin-Infused Relaxed Algorithm (MIRA) (Crammer et al., 2006). This model computes the score of a parse tree as the inner product of a feature vector and a weight vector representing model parameters. The training procedure of MIRA is very close to that of a perceptron (Rosenblatt, 1958), benefiting from its speed and relatively low requirements while achieving better accuracy.

Recall that parsing under this model consists in (1) generating a n-best list of constituency parses using the generative model, (2) annotating each of them with function tags, (3) converting them to dependency parses, (4) extracting features, (5) scoring each feature vector against the model, (6) selecting the highest scoring parse as output.

For training, we collect the output of feature extraction (4) for a large set of training sentences and associate each parse tree with a loss function that denotes the number of erroneous dependencies compared to the reference parse tree. Then, model weights are adjusted using MIRA training so that the parse with the lowest loss gets the highest score. Examples are processed in sequence, and for each of them, we compute the score of each parse according to the current model and find an updated weight vector that assigns the first rank to the best parse (called *oracle*). Details of the algorithm are given in the following sections.

3.1 Definitions

Let us consider a vector space of dimension m where each component corresponds to a feature: a parse tree p is represented as a sparse vector $\phi(p)$. The model is a weight vector w in the same space where each weight corresponds to the importance of the features for characterizing good (or bad) parse trees. The score $s(p)$ of a parse tree p is the scalar product of its feature vector $\phi(p)$ and the weight vector w.

$$s(p) = \sum_{i=1}^{m} w_i \phi_i(p) \tag{2}$$

Let L be the n-best list of parses produced by the generative parser for a given sentence. The highest scoring parse $\hat{p}$ is selected as output of the reranker:

$$\hat{p} = \underset{p \in L}{\text{argmax}}\, s(p) \tag{3}$$

MIRA learning consists in using training sentences and their reference parses to determine the weight vector w. It starts with $w = 0$ and modifies it incrementally so that parses closest to the reference get higher scores. Let $l(p)$, loss of parse p, be the number of erroneous dependencies (governor, dependent, label) compared to the reference parse. We define o, the oracle parse, as the parse with the lowest loss in L.

Training examples are processed in sequence as an instance of online learning. For each sentence, we compute the score of each parse in the n-best list. If the highest scoring parse differs from the oracle ($\hat{p} \neq o$), the weight vector can be improved. In this case, we seek a modification of w ensuring that o gets a better score than $\hat{p}$ with a difference at least proportional to the difference between their loss. This way, very bad parses get pushed deeper than average parses. Finding such weight vector can be formulated as the following constrained optimization problem:

$$\text{minimize: } ||w||^2 \tag{4}$$

$$\text{subject to: } s(o) - s(\hat{p}) \geq l(o) - l(\hat{p}) \tag{5}$$

Since there is an infinity of weight vectors that satisfy constraint 5, we settle on the one with the smallest magnitude. Classical constrained quadratic optimization methods can be applied to solve this problem: first, Lagrange multipliers are used to introduce the constraint in the objective function, then the Hildreth algorithm yields the following analytic solution to the non-constrained problem:

$$w^\star = w + \alpha\,(\phi(o) - \phi(\hat{p})) \tag{6}$$

$$\alpha = \max\left[0, \frac{l(o) - l(\hat{p}) - [s(o) - s(\hat{p})]}{||\phi(o) - \phi(\hat{p})||^2}\right] \tag{7}$$

Here, $w^\star$ is the new weight vector, α is an update magnitude and $[\phi(o) - \phi(\hat{p})]$ is the difference between the feature vector of the oracle and that of the highest scoring parse. This update, similar to the perceptron update, draws the weight vector towards o while pushing it away from $\hat{p}$. Usual tricks that apply to the perceptron also apply here: (a) performing multiple passes on the training data, and (b) averaging the weight vector over each update[2]. Algorithm 1 details the instructions for MIRA training.

Algorithm 1 MIRA training

```
for i = 1 to t do
  for all sentences in training set do
    Generate n-best list L from generative parser
    for all p ∈ L do
      Extract feature vector φ(p)
      Compute score s(p) (eq. 2)
    end for
    Get oracle o = argmin_p l(p)
    Get best parse p̂ = argmax_p s(p)
    if p̂ ≠ o then
      Compute α (eq. 7)
      Update weight vector (eq. 6)
    end if
  end for
end for
Return average weight vector over updates.
```

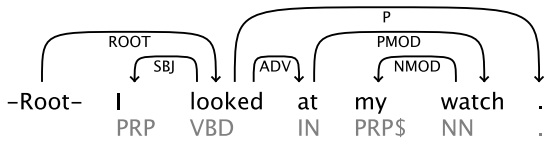
3.2 Features

The quality of the reranker depends on the learning algorithm as much as on the feature set. These features can span over any subset of a parse tree, up to the whole tree. Therefore, there are a very large set of possible features to choose from. Relevant features must be general enough to appear in as many

[2]This can be implemented efficiently using two weight vectors as for the averaged perceptron.

parses as possible, but specific enough to characterize good and bad configurations in the parse tree.

We extended the feature set from (McDonald, 2006) which showed to be effective for a range of languages. Our feature templates can be categorized in 5 classes according to their domain of locality. In the following, we describe and exemplify these templates on the following sentence from the Penn treebank, in which we target the PMOD dependency between "at" and "watch."



Probability Three features are derived from the PCFG-LA parser, namely the posterior probability of the parse (eq. 1), its normalized probability relative to the 1-best, and its rank in the n-best list.

Unigram Unigram features are the most simple as they only involve one word. Given a dependency between position i and position j of type l, governed by x_i, denoted $x_i \xrightarrow{l} x_j$, two features are created: one for the governor x_i and one for the dependent x_j. They are described as 6-tuples (word, lemma, pos-tag, is-governor, direction, type of dependency). Variants with wildcards at each subset of tuple slots are also generated in order to handle sparsity.

In our example, the dependency between "looked" and "at" generates two features:

`[at, at, IN, G, R, PMOD]` and `[looked, look, NN, D, L, PMOD]`

And also wildcard features such as:

`[-, at, IN, G, R, PMOD], [at, -, IN, G, R, PMOD] ...`
`[at, -, -, -, -, PMOD]`

This wildcard feature generation is applied to all types of features. We will omit it in the remainder of the description.

Bigram Unlike the previous template, bigram features model the conjunction of the governor and the dependent of a dependency relation, like bilexical dependencies in (Collins, 1997). Given dependency $x_i \xrightarrow{l} x_j$, the feature created is (word x_i, lemma x_i, pos-tag x_i, word x_j, lemma x_j, pos-tag x_j, distance[3] from i to j, direction, type).

The previous example generates the following feature:

`[at, at, IN, watch, watch, NN, 2, R, PMOD]`

Where 2 is the distance between "at" and "watch".

Linear context This feature models the linear context between the governor and the dependent of a relation by looking at the words between them. Given dependency $x_i \xrightarrow{l} x_j$, for each word from $i+1$ to $j-1$, a feature is created with the pos-tags of x_i and x_j, and the pos tag of the word between them (no feature is create if $j = i+1$). An additional feature is created with pos-tags at positions $i-1, i, i+1, j-1, j, j+1$. Our example yields the following features:

`[`**IN**`, PRP$, `**NN**`]`, and `[VBD, `**IN**`, PRP$, PRP$, `**NN**`, .]`.

Syntactic context: siblings This template and the next one look at two dependencies in two configurations. Given two dependencies $x_i \xrightarrow{l} x_j$ and $x_i \xrightarrow{m} x_k$, we create the feature (word, lemma, pos-tag for x_i, x_j and x_k, distance from i to j, distance from i to k, direction and type of each of the two dependencies). In our example:

`[looked, look, VBD, I, I, PRP, at, at, IN, 1, 1, L, SBJ, R, ADV]`

Syntactic context: chains Given two dependencies $x_i \xrightarrow{l} x_j \xrightarrow{m} x_k$, we create the feature (word, lemma, pos-tag of x_i, x_j and x_k, distance from i to j, distance from i to k, direction and type of each of the two dependencies). In our example:

`[looked, look, VBD, at, at, IN, watch, watch, NN, 1, 2, R, ADV,`

[3] In every template, distance features are quantified in 7 classes: 1, 2, 3, 4, 5, 5 to 10, more.

R, PMOD]

It is worth noting that our feature templates only rely on information available in the training set, and do not use any external linguistic knowledge.

4 Experiments

In this section, we evaluate our architecture on two corpora, namely the Penn Treebank (Marcus et al., 1994) and the French Treebank (Abeillé et al., 2003). We first present the corpora and the tools used for annotating and converting structures, then the performances of the phrase structure parser alone and with the discriminative reranker.

4.1 Treebanks and Tools

For English, we use the Wall Street Journal sections of the Penn Treebank. We learn the PCFG-LA from sections 02-21[4]. We then use FUNTAG (Chrupała et al., 2007) to add functions back to the PCFG-LA analyses. For the conversion to dependency structures we use the LTH tool (Johansson and Nugues, 2007). In order to get the gold dependencies, we run LTH directly on the gold parse trees. We use section 22 for development and section 23 for the final evaluation.

For French, we use the Paris 7 Treebank (or French Treebank, FTB). As in several previous experiments we decided to divide the 12,350 phrase structure trees in three sets: train (80%), development (10%) and test (10%). The syntactic tag set for French is not fixed and we decided to use the one described in (Candito and Crabbé, 2009) to be able to compare this system with recent parsing results on French. As for English, we learn the PCFG-LA without functional annotations which are added afterwards. We use the dependency structures developed in (Candito et al., 2010b) and the conversion toolkit BONSAÏ. Furthermore, to test our approach against state of the art parsing results for French we use word clusters in the phrase-based parser as in (Candito and Crabbé, 2009).

For both languages we constructed 10-fold training data from train sets in order to avoid overfitting the training data. The trees from training sets were divided into 10 subsets and the parses for each subset were generated by a parser trained on the other 9 subsets. Development and test parses are given by a parser using the whole training set. Development sets were used to choose the best reranking model.

For lemmatisation, we use the MATE lemmatiser for English and a home-made lemmatiser for French based on the le*fff* lexicon (Sagot, 2010).

4.2 Generative Model

The performances of our parser are summarised in Figure 2, (a) and (b), where F-score denotes the Parseval F-score[5], and LAS and UAS are respectively the Labelled and Unlabelled Attachment Score of the converted dependency structures[6]. We give oracle scores (the score that our system would get if it selected the best parse from the n-best lists) when the parser generates n-best lists of depth 10, 20, 50 and 100 in order to get an idea of the effectiveness of the reranking process.

One of the issues we face with this approach is the use of an imperfect functional annotator. For French we evaluate the loss of accuracy on the resulting dependency structure from the gold development set where functions have been omitted. The UAS is 100% but the LAS is 96.04%. For English the LAS from section 22 where functions are omitted is 95.35%.

From the results presented in this section we can make two observations. First, the results of our parser are at the state of the art on English (90.7% F-score) and on French (85.7% F-score). So the reranker will be confronted with the difficult task of improving on these scores. Second, the progression margin is sensible with a potential LAS error reduction of 41% for English and 40.2% for French.

4.3 Adding the Reranker

4.3.1 Learning Feature Weights

The discriminative model, i.e. template instances and their weights, is learnt on the training set parses obtained via 10-fold cross-validation. The generative parser generates 100-best lists that are used as learning example for the MIRA algorithm. Feature extraction produces an enormous number of features: about 571 millions for English and 179 mil-

[4]Functions are omitted.

[5]We use a modified version of `evalb` that gives the oracle score when the parser outputs a list of candidates for each sentence.

[6]All scores are measured without punctuation.

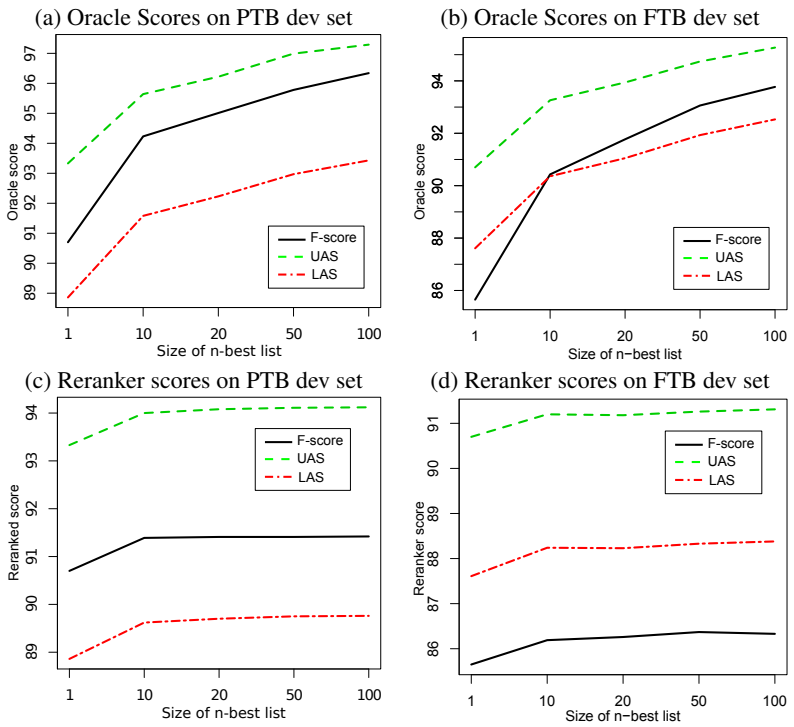


Figure 2: Oracle and reranker scores on PTB and FTB data on the dev. set, according to the depth of the n-best.

lions for French. Let us remark that this large set of features is not an issue because our discriminative learning algorithm is online, that is to say it considers only one example at a time, and it only gives non-null weights to useful features.

4.3.2 Evaluation

In order to test our system we first tried to evaluate the impact of the length of the n-best list over the reranking predictions[7]. The results are shown in Figure 2, parts (c) and (d).

For French, we can see that even though the LAS and UAS are consistently improving with the number of candidates, the F-score is maximal with 50 candidates. However the difference between 50 candidates and 100 candidates is not statistically significant. For English, the situation is simpler and scores improve continuously on the three metrics.

Finally we run our system on the test sets for both treebanks. Results are shown[8] in Table 1 for English, and Table 2 for French. For English the improvement is 0.9% LAS, 0.7% Parseval F-score and 0.8% UAS.

	Baseline	Reranker
F	90.4	91.1
F < 40	91.0	91.7
LAS	88.9	89.8
UAS	93.1	93.9

Table 1: System results on PTB Test set

For French we have improvements of 0.3/0.7/0.9. If we add a template feature indicating the agreement between part-of-speech provided by the PCFG-LA parser and a part-of-speech tagger (Denis and Sagot, 2009), we obtain better improvements: 0.5/0.8/1.1.

	Baseline	Reranker	Rerank + MElt
F	86.6	87.3	87.4
F < 40	88.7	89.0	89.2
LAS	87.9	89.0	89.2
UAS	91.0	91.9	92.1

Table 2: System results on FTB Test set

[7] The model is always trained with 100 candidates.

[8] F < 40 is the parseval F-score for sentences with less than 40 words.

4.3.3 Comparison with Related Work

We compare our results with related parsing results on English and French.

For English, the main results are shown in Table 3. From the presented data, we can see that indirect reranking on LAS may not seem as good as direct reranking on phrase-structures compared to F-scores obtained in (Charniak and Johnson, 2005) and (Huang, 2008) with one parser or (Zhang et al., 2009) with several parsers. However, our system does not rely on any language specific feature and can be applied to other languages/treebanks. It is difficult to compare our system for LAS because most systems evaluate on gold data (part-of-speech, lemmas and morphological information) like Bohnet (2010).

Our system also compares favourably with the system of Carreras et al. (2008) that relies on a more complex generative model, namely Tree Adjoining Grammars, and the system of Suzuki et al. (2009) that makes use of external data (unannotated text).

	F	LAS	UAS
Huang, 2008	91.7	–	–
Bohnet, 2010	–	90.3	–
Zhang et al, 2008	91.4	–	93.2
Huang and Sagae, 2010	–	–	92.1
Charniak et al, 2005	91.5	90.0	94.0
Carreras et al. 2008	–	–	93.5
Suzuki et al. 2009	–	–	93.8
This work	91.1	89.8	93.9

Table 3: Comparison on PTB Test set

For French, see Table 4, we compare our system with the MATE parser (Bohnet, 2010), an improvement over the MST parser (McDonald et al., 2005) with hash kernels, using the MELT part-of-speech tagger (Denis and Sagot, 2009) and our own lemmatiser.

We also compare the French system with results drawn from the benchmark performed by Candito et al. (2010a). The first system (BKY-FR) is close to ours without the reranking module, using the Berkeley parser adapted to French. The second (MST-FR) is based on MSTParser (McDonald et al., 2005). These two system use word clusters as well.

The next section takes a close look at the models of the reranker and its impact on performance.

	F < 40	LAS	UAS
This work	**89.2**	**89.2**	**92.1**
MATE + MELT	–	89.2	91.8
BKY-FR	88.2	86.8	91.0
MST-FR	–	88.2	90.9

Table 4: Comparison on FTB Test set

4.3.4 Model Analysis

It is interesting to note that in the test sets, the one-best of the syntagmatic parser is selected 52.0% of the time by the reranker for English and 34.3% of the time for French. This can be explained by the difference in the quantity of training data in the two treebanks (four times more parses are available for English) resulting in an improvement of the quality of the probabilistic grammar.

We also looked at the reranking models, specifically at the weight given to each of the features. It shows that 19.8% of the 571 million features have a non-zero weight for English as well as 25.7% of the 179 million features for French. This can be explained by the fact that for a given sentence, features that are common to all the candidates in the n-best list are not discriminative to select one of these candidates (they add the same constant weight to the score of all candidates), and therefore ignored by the model. It also shows the importance of feature engineering: designing relevant features is *an art* (Charniak and Johnson, 2005).

We took a closer look at the 1,000 features of highest weight and the 1,000 features of lowest weight (negative) for both languages that represent the most important features for discriminating between correct and incorrect parses. For English, 62.0% of the positive features are backoff features which involve at least one wildcard while they are 85.9% for French. Interestingly, similar results hold for negative features. The difference between the two languages is hard to interpret and might be due in part to lexical properties and to the fact that these features may play a balancing role against towards non-backoff features that promote overfitting.

Expectedly, posterior probability features have the highest weight and the n-best rank feature has the highest negative weight. As evidenced by Table 5,

	en (+)	en (-)	fr (+)	fr (-)
Linear	30.4	36.1	44.8	44.0
Unigram	20.7	16.3	9.7	8.2
Bigram	27.4	29.1	20.8	24.4
Chain	15.4	15.3	13.7	19.4
Siblings	5.8	3.0	10.8	3.6

Table 5: Repartition of weight (in percentage) in the 1,000 highest (+) and lowest (-) weighted features for English and French.

among the other feature templates, linear context occupies most of the weight mass of the 1,000 highest weighted features. It is interesting to note that the unigram and bigram templates are less present for French than for English while the converse seems to be true for the linear template. Sibling features are consistently less relevant.

In terms of LAS performance, on the PTB test set the reranked output is better than the baseline on 22.4% of the sentences while the opposite is true for 10.4% of the sentences. In 67.0% of the sentences, they have the same LAS (but not necessarily the same errors). This emphasises the difficulty of reranking an already good system and also explains why oracle performance is not reached. Both the baseline and reranker output are completely correct on 21.3% of the sentences, while PCFG-LA correctly parses 23% of the sentences and the MIRA brings that number to 26%.

Figures 3 and 4 show hand-picked sentences for which the reranker selected the correct parse. The French sentence is a typical difficult example for PCFGs because it involves a complex rewriting rule which might not be well covered in the training data (SENT $\rightarrow$ NP VP PP PONCT PP PONCT PP PONCT). The English example is tied to a wrong attachment of the prepositional phrase to the verb instead of the date, which lexicalized features of the reranker handle easily.

5 Conclusion

We showed that using a discriminative reranker, on top of a phrase structure parser, based on converted dependency structures could lead to significant improvements over dependency and phrase structure parse results. We experimented on two treebanks for two languages, English and French and we measured the improvement of parse quality on three different metrics: Parseval F-score, LAS and UAS, with the biggest error reduction on the latter. However the gain is not as high as expected by looking at oracle scores, and we can suggest several possible improvements on this method.

First, the sequential approach is vulnerable to cascading errors. Whereas the generative parser produces several candidates, this is not the case of the functional annotators: these errors are not amendable. It should be possible to have a functional tagger with ambiguous output upon which the reranker could discriminate. It remains an open question as how to integrate ambiguous output from the parser and from the functional tagger. The combination of n-best lists would not scale up and working on the ambiguous structure itself, the packed forest as in (Huang, 2008), might be necessary. Another possibility for future work is to let the phrase-based parser itself perform function annotation, but some preliminary tests on French showed disappointing results.

Second, designing good features, sufficiently general but precise enough, is, as already coined by Charniak and Johnson (2005), *an art*. More formally, we can see several alternatives. Dependency structures could be exploited more thoroughly using, for example, tree kernels. The restricted number of candidates enables the use of more global features. Also, we haven't used any language-specific syntactic features. This could be another way to improve this system, relying on external linguistic knowledge (lexical preferences, subcategorisation frames, copula verbs, coordination symmetry ...). Integrating features from the phrase-structure trees is also an option that needs to be explored.

Third this architecture enables the integration of several systems. We experimented on French using a part-of-speech tagger but we could also use another parser and either use the methodology of (Johnson and Ural, 2010) or (Zhang et al., 2009) which fusion n-best lists form different parsers, or use stacking methods where an additional parser is used as a guide for the main parser (Nivre and McDonald, 2008).

Finally it should be noted that this system does not rely on any language specific feature, and thus can be applied to languages other that French or English

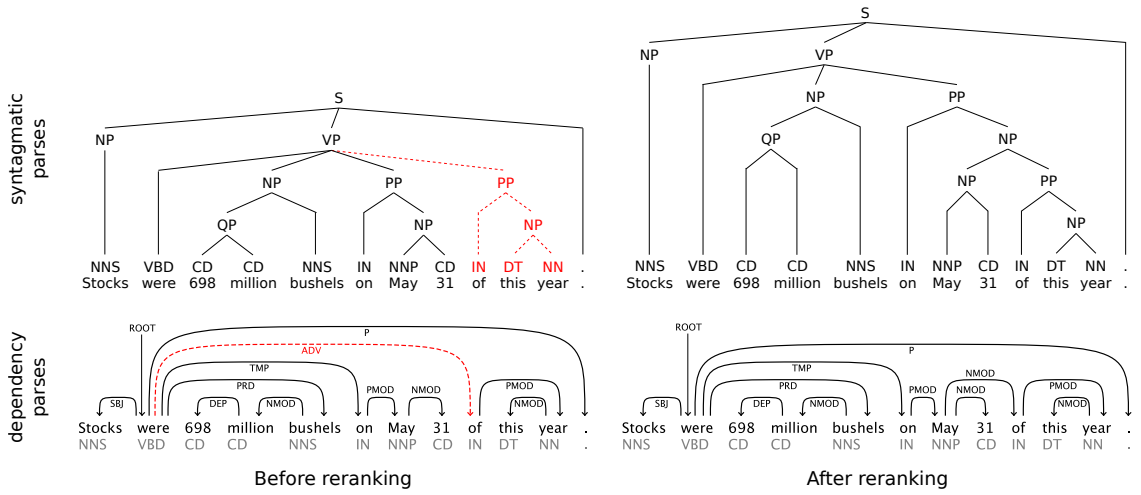


Figure 3: English sentence from the WSJ test set for which the reranker selected the correct tree while the first candidate of the n-best list suffered from an incorrect attachment.

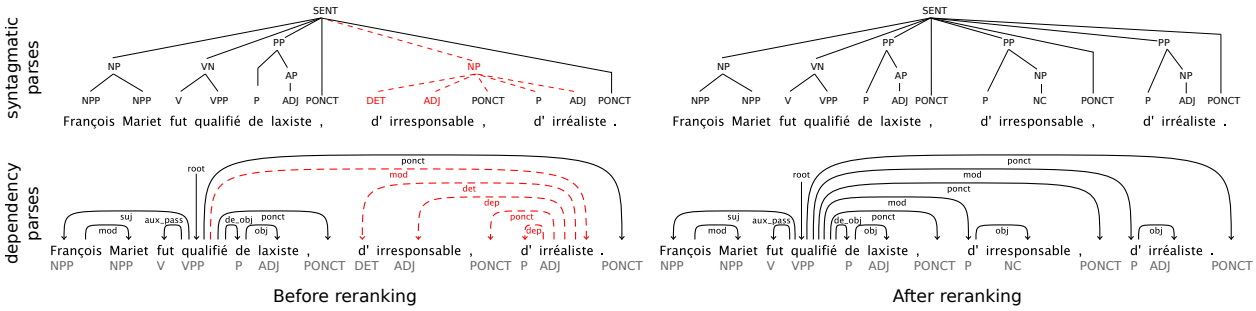


Figure 4: Sentence from the FTB for which the best parse according to baseline was incorrect, probably due to the tendency of the PCFG-LA model to prefer rules with more support. The reranker selected the correct parse.

without re-engineering new reranking features. This makes this architecture suitable for morphologically rich languages.

Acknowledgments

This work has been funded by the French Agence Nationale pour la Recherche, through the project SEQUOIA (ANR-08-EMER-013).

References

Anne Abeillé, Lionel Clément, and Toussenel François, 2003. *Treebanks*, chapter Building a treebank for French. Kluwer, Dordrecht.

M. Attia, J. Foster, D. Hogan, J. Le Roux, L. Tounsi, and J. van Genabith. 2010. Handling Unknown Words in Statistical Latent-Variable Parsing Models for Arabic, English and French. In *Proceedings of SPMRL*.

Bernd Bohnet. 2010. Top Accuracy and Fast Dependency Parsing is not a Contradiction. In *Proceedings of COLING*.

M.-H. Candito and B. Crabbé. 2009. Improving Generative Statistical Parsing with Semi-Supervised Word Clustering. In *Proceedings of IWPT 2009*.

M.-H. Candito, J. Nivre, P. Denis, and E. Henestroza Anguiano. 2010a. Benchmarking of Statistical Dependency Parsers for French. In *Proceedings of COLING'2010*.

Marie Candito, Benoît Crabbé, and Pascal Denis. 2010b. Statistical French Dependency Parsing : Treebank Conversion and First Results. In *Proceedings of LREC2010*.

Xavier Carreras, Michael Collins, and Terry Koo. 2008. TAG, Dynamic Programming and the Perceptron for Efficient, Feature-rich Parsing. In *CONLL*.

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

Grzegorz Chrupała, Nicolas Stroppa, Josef van Genabith, and Georgiana Dinu. 2007. Better training for function labeling. In *Proceedings of RANLP*, Borovets, Bulgaria.

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*.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai ShalevShwartz, and Yoram Singer. 2006. Online Passive-Aggressive Algorithm. *Journal of Machine Learning Research*.

Pascal Denis and Benoît Sagot. 2009. Coupling an annotated corpus and a morphosyntactic lexicon for state-of-the-art pos tagging with less human effort. In *Proceedings PACLIC 23*, Hong Kong, China.

Liang Huang. 2008. Forest Reranking: Discriminative Parsing with Non-Local Features. In *Proceedings of ACL*.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Mark Johnson and Ahmet Engin Ural. 2010. Reranking the Berkeley and Brown Parsers. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 665–668, Los Angeles, California, June. Association for Computational Linguistics.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the ARPA Speech and Natural Language Workshop*.

Takuya Matsuzaki, Yusuke Miyao, and Jun ichi Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of ACL*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005. Online Large-Margin Training of Dependency Parsers. In *Association for Computational Linguistics (ACL)*.

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

J. Nivre and R. McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of ACL*, pages 950–958.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *HLT-NAACL*, pages 404–411.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *ACL*.

Geoffrey K. Pullum and Barbara C. Scholz. 2001. On the distinction between model-theoretic and generative-enumerative syntactic frameworks. In *Logical Aspects of Computational Linguistics*.

Owen Rambow. 2010. The Simple Truth about Dependency and Phrase Structure Representations: An Opinion Piece. In *NAACL HLT*.

Frank Rosenblatt. 1958. The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain. *Psychological Review*.

Benoît Sagot. 2010. The lefff, a freely available and large-coverage lexicon for french. In *Proceedings of LREC 2010, La Valette, Malta*.

J. Suzuki, H. Isozaki, X. Carreras, and M. Collins. 2009. An empirical study of semi-supervised structured conditional models for dependency parsing. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 551–560. Association for Computational Linguistics.

Hui Zhang, Min Zhang, Chew Lim Tan, and Haizhou Li. 2009. K-best combination of syntactic parsers. In *Proceedings of EMNLP*.