A generative re-ranking model for dependency parsing

Federico Sangati, Willem Zuidema and Rens Bod
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 904, 1098 XH Amsterdam, The Netherlands
{f.sangati,zuidema,rens.bod}@uva.nl

Abstract

We propose a framework for dependency parsing based on a combination of discriminative and generative models. We use a discriminative model to obtain a *k*-best list of candidate parses, and subsequently rerank those candidates using a generative model. We show how this approach allows us to evaluate a variety of generative models, without needing different parser implementations. Moreover, we present empirical results that show a small improvement over state-of-the-art dependency parsing of English sentences.

1 Introduction

Probabilistic generative dependency models define probability distributions over all valid dependency structures, and thus provide a useful intermediate representation that can be used for many NLP tasks including parsing and language modeling. In recent evaluations of supervised dependency parsing, however, generative approaches are consistently outperformed by discriminative models (Buchholz et al., 2006; Nivre et al., 2007), which treat the task of assigning the correct structure to a given sentence as a classification task. In this category we include both transition based (Nivre and Hall , 2005) and graph based parsers (McDonald, 2006).

In this paper, we explore a reranking approach that combines a generative and a discrimative model and tries to retain the strengths of both. The idea of combining these two types of models through re-ranking is not new, although it has been mostly explored in constituency parsing (Collins et al., 2002). This earlier work, however, used the generative model in the first step, and trained the discriminative model over its *k*-best candidates. In this paper we reverse the usual order of the two models, by employing a generative model to rescore the *k*-best candidates provided by a discriminative model. Moreover, the generative model of the second phase uses frequency counts from the training set but is not trained on the *k*-best parses of the discriminative model.

The main motivation for our approach is that it allows for efficiently evaluating many generative models, differing from one another on (i) the choice of the linguistic units that are generated (words, pairs of words, word graphs), (ii) the generation process (Markov process, top-down, bottom-up), and (iii) the features that are considered to build the event space (postags/words, distance). Although efficient algorithms exist to calculate parse forests (Eisner, 1996a), each choice gives rise to different parser instantiations.

1.1 A generative model for re-ranking

In our re-ranking perspective, all the generative model has to do is to compute the probability of *k* pre-generated structures, and select the one with maximum probability. In a generative model, every structure can be decomposed into a series of independent events, each mapped to a corresponding conditioning event. As an example, if a generative model chooses D as the right dependent of a certain word H, conditioned uniquely on their relative position, we can define the event as D *is the right dependent of* H, and the conditioning event as H *has a right dependent*.

As a preprocessing step, every sentence structure in the training corpus is decomposed into a series of independent events, with their corresponding conditioning events. During this process, our model updates two tables containing the frequency of events and their conditioning counterparts.

In the re-ranking phase, a given candidate structure can be decomposed into independent events $(e_1, e_2, \ldots, e_n)$ and corresponding conditioning events $(c_1, c_2, \ldots, c_n)$ as in the training phase.

Proceedings of the 11th International Conference on Parsing Technologies (IWPT), pages 238–241,
Paris, October 2009.

The probability of the structure can then be calculated as

$$\prod_{i=1}^{n} \frac{f(e_i)}{f(c_i)} \quad (1)$$

where $f(x)$ returns the frequency of x previously stored in the tables.

It is important to stress the point that the only specificity each generative model introduces is in the way sentence structures are decomposed into events; provided a generic representation for the (conditioning) event space, both training phase and probability calculation of candidate structures can be implemented independently from the specific generative model, through the implementation of generic tables of (conditioning) events.

In this way the probabilities of candidate structures are exact probabilities, and do not suffer from possible approximation techniques that parsers often utilize (i.e., pruning). On the other hand the most probable parse is selected from the set of the *k* candidates generated by the discriminative model, and it will equal with the most probable parse among all possible structures, only for sufficiently high *k*.

2 MST discriminative model

In order to generate a set of k-candidate structures for every test sentence, we use a state-of-the-art discriminative model (McDonald, 2006). This model treats every dependency structure as a set of word-dependent relations, each described by a high dimensional feature representation. For instance, if in a certain sentence word i is the head of word j, $\mathbf{v}(i,j)$ is the vector describing all the features of such relation (i.e., labels of the two words, their postag, and other information including words in between them, and ancestral nodes). During the training phase the model learns a weight vector $\mathbf{w}$ which is then used to find the best dependency structure y for a given test sentence x. The score that needs to be maximized is defined as $\sum_{(i,j)\in y} \mathbf{w} \cdot \mathbf{v}(i,j)$, and the best candidate is called the maximum spanning tree (MST).

Assuming we have the weight vector and we only consider projective dependency structures, the search space can be efficiently computed by using a dynamic algorithm on a compact representation of the parse forest (Eisner, 1996a). The training phase is more complex; for details we refer to (McDonald, 2006). Roughly, the model employs a large-margin classifier which iterates over the structures of the training corpus, and updates the weight vector $\mathbf{w}$ trying to keep the score of the correct structure above the scores of the incorrect ones by an amount which is proportional to how much they differ in accuracy.

3 Generative model

3.1 Eisner model

As a generative framework we have chosen to use a variation of model C in (Eisner, 1996a). In this approach nodes are generated recursively in a top-down manner starting from the special symbol *EOS* (end of sentence). At any given node, left and right children are generated as two separate Markov sequences of nodes[1], each conditioned on ancestral and sibling information (which, for now, we will simply refer to as *context*).

One of the relevant variations with respect to the original model is that in our version the direction of the Markov chain sequence is strictly left to right, instead of the usual inside outwards.

More formally, given a dependency structure T, and any of its node N, the probability of generating the fragment $T(N)$ of the dependency structure rooted in N is defined as:

$$P(T(N)) = \prod_{l=1}^{L} P(N_{\lhd l})|context) \cdot P(T(N_{\lhd l})) \times \prod_{r=1}^{R} P(N_{\rhd r})|context) \cdot P(T(N_{\rhd r})) \quad (2)$$

where L and R are the number of left and right children of N in T $(L, R > 0)$, $N_{\lhd l}$ is the left daughter of N at position l in T (analogously for right daughters). The probability of the entire dependency structure T is computed as $P(T(\textit{EOS}))$.

In order to illustrate how a dependency structure can be decomposed into events, we present in table 1 the list of events and the corresponding conditioning events extracted from the dependency structure illustrated in figure 1. In this simple example, each node is identified with its word, and the context is composed of the direction with respect to the head node, the head node, and the previously chosen daughter (or *NONE* if it is the first). While during the training phase the event tables are updated with these events, in the test phase they are looked-up to compute the structure probability, as in equation 1.

[1] Every sequence ends with the special symbol *EOC*.

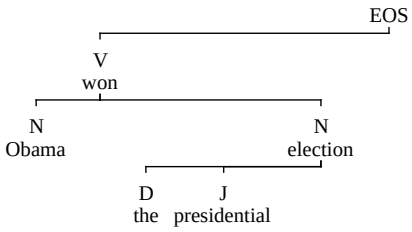

Figure 1: Dependency tree of the sentence *"Obama won the presidential election"*.

3.2 Model extension

In equation 2 we have generically defined the probability of choosing a daughter D based on specific features associated with D and the context in which it occurs. In our implementation, this probability is instantiated as in equation 3. The specific features associated with D are: the distance[2] $dist(H, D)$ between D and its head H, the flag $term(D)$ which specifies whether D has more dependents, and the lexical and postag representation of D. The context in which D occurs is defined by features of the head node H, the previously chosen sister S, the grandparent G, and the direction dir (left or right).

Equation 3 is factorized in four terms, each employing an appropriate backoff reduction list reported in descending priority[3].

$$
\begin{aligned}
&P(D|context) = \qquad (3)\\
&P(dist(H,D), term(D), word(D), tag(D)|H,S,G,dir) =\\
&P(tag(D)|H,S,G,dir)\\
&\quad \text{reduction list: } \begin{array}{|l|}\hline wt(H), wt(S), wt(G), dir\\ \hline wt(H), wt(S), t(G), dir\\ \hline \left\{\begin{array}{l} wt(H), t(S), t(G), dir\\ t(H), wt(S), t(G), dir\end{array}\right.\\ \hline t(H), t(S), t(G), dir\\ \hline\end{array}\\
&\times P(word(D)|tag(D), H, S, G, dir)\\
&\quad \text{reduction list: } \begin{array}{|l|}\hline wt(H), t(S), dir\\ \hline t(H), t(S), dir\\ \hline\end{array}\\
&\times P(term(D)|word(D), tag(D), H, S, G, dir)\\
&\quad \text{reduction list: } \begin{array}{|l|}\hline tag(D), wt(H), t(S), dir\\ \hline tag(D), t(H), t(S), dir\\ \hline\end{array}\\
&\times P(dist(P,D)|term(D), word(D), tag(D), H, S, G, dir)\\
&\quad \text{reduction list: } \begin{array}{|l|}\hline word(D), tag(D), t(H), t(S), dir\\ \hline tag(D), t(H), t(S), dir\\ \hline\end{array}
\end{aligned}
$$

[2]In our implementation distance values are grouped in 4 categories: $1, 2, 3-6, 7-\infty$.

[3]In the reduction lists, $wt(N)$ stands for the string incorporating both the postag and the word of N, and $t(N)$ stands for its postag. This second reduction is never applied to closed class words. All the notation and backoff parameters are identical to (Eisner, 1996b), and are not reported here for reasons of space.

[4]The counts are extracted from a two-sentence corpus which also includes "Obama lost the election."

Events	**Freq.**	**Conditioning Events**	**Freq.**
won L *EOS NONE*	1	L *EOS NONE*	2
EOC L *EOS* won	1	L *EOS* won	1
EOC R *EOS NONE*	2	R *EOS NONE*	2
Obama L won *NONE*	1	L won *NONE*	1
EOC L won Obama	1	L won Obama	1
election R won *NONE*	1	R won *NONE*	1
EOC R won election	1	R won election	1
EOC L Obama *NONE*	2	L Obama *NONE*	2
EOC R Obama *NONE*	2	R Obama *NONE*	2
the L election *NONE*	2	L election *NONE*	2
presidential L election the	1	L election the	2
EOC L election presidential	1	L election presidential	1
EOC R election *NONE*	2	R election *NONE*	2
EOC L the *NONE*	2	L the *NONE*	2
EOC R the *NONE*	2	R the *NONE*	2
EOC L presidential *NONE*	1	L presidential *NONE*	1
EOC R presidential *NONE*	1	R presidential *NONE*	1

Table 1: Events occurring when generating the dependency structure in figure 1, for the event space (dependent | direction, head, sister). According to the reported frequency counts[4], the structure has a associated probability of $1/4$.

4 Results

In our investigation, we have tested our model on the Wall Street Journal corpus (Marcus et al., 1993) with sentences up to 40 words in length, converted to dependency structures. Although several algorithms exist to perform such a conversion (Sangati and Zuidema, 2008), we have followed the scheme in (Collins, 1999). Section 2-21 was used as training, and section 22 as test set. The MST discriminative parser was provided with the correct postags of the words in the test set, and it was run in second-order[5] and projective mode. Results are reported in table 2, as unlabeled attachment score (UAS). The MST dependency parser obtains very high results when employed alone (92.58%), and generates a list of *k*-best-candidates which can potentially achieve much better results (an oracle would score above 95% when selecting from the first 5-best, and above 99% from the first 1000-best). The decrease in performance of the generative model, as the number of the candidate increases, suggests that its performance would be lower than a discriminative model if used alone. On the other hand, our generative model is able to select better candidates than the MST parser, when their number is limited to a few dozens, yielding a maximum accuracy for $k = 7$ where it improves accuracy on the discriminative model by a 0.51% (around 7% error reduction).

[5]The features of every dependency relation include information about the previously chosen sister of the dependent.

k-best	Oracle best	Oracle worst	Reranked
1	**92.58**	92.58	92.58
2	94.22	88.66	92.89
3	95.05	87.04	93.02
4	95.51	85.82	93.02
5	95.78	84.96	93.02
6	96.02	84.20	93.06
7	96.23	83.62	**93.09**
8	96.40	83.06	93.02
9	96.54	82.57	92.97
10	96.64	82.21	92.96
100	98.48	73.30	92.32
1000	99.34	64.86	91.47

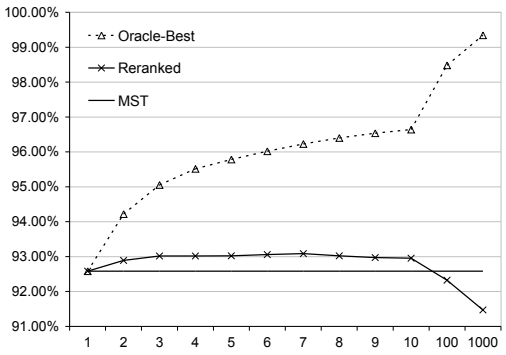

Figure 2: UAS accuracy of the MST discriminative and re-ranking parser on section 22 of the WSJ. *Oracle best*: always choosing the best result in the *k*-best, *Oracle worst*: always choosing the worst, *Reranked*: choosing the most probable candidate according to the generative model.

5 Conclusions

We have presented a general framework for dependency parsing based on a combination of discriminative and generative models. We have used this framework to evaluate and compare several generative models, including those of Eisner (1996) and some of their variations. Consistently with earlier results, none of these models performs better than the discriminative baseline when used alone. We have presented an instantiation of this framework in which our newly defined generative model leads to an improvement of the state-of-the-art parsing results, when provided with a limited number of best candidates. This result suggests that discriminative and generative model are complementary: the discriminative model is very accurate to filter out "bad" candidates, while the generative model is able to further refine the selection among the few best candidates. In our set-up it is now possible to efficiently evaluate many other generative models and identify the most promising ones for further investigation. And even though we currently still need the input from a discriminative model, our promising results show that pessimism about the prospects of probabilistic generative dependency models is premature.

Acknowledgments We gratefully acknowledge funding by the Netherlands Organization for Scientific Research (NWO): FS and RB are funded through a Vici-grant "Integrating Cognition" (277.70.006) to RB, and WZ through a Veni-grant "Discovering Grammar" (639.021.612) of NWO. We also thank 3 anonymous reviewers for useful comments.

References

S. Buchholz, and E. Marsi. 2006. CoNLL-X Shared Task on Multilingual Dependency Parsing. In *Proc. of the 10th CoNLL Conference*, pp. 149–164.

M. Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

M. Collins, N. Duffy, and F. Park. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *In Proceedings of the ACL 2002*, pp. 263–270.

J. Eisner. 1996a. Three New Probabilistic Models for Dependency Parsing: An Exploration. In *Proc. of the 16th International Conference on Computational Linguistics (COLING-96)*, pp. 340–345.

J. Eisner. 1996b. An Empirical Comparison of Probability Models for Dependency Grammar. *Technical Report number IRCS-96-11*, Univ. of Pennsylvania.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. In *Computational Linguistics*, 19(2), pp. 313–330.

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

J. Nivre and J. Hall. 2005. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. In *Proc. of the Fourth Workshop on Treebanks and Linguistic Theories*, pp. 137–148.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson,S. Riedel, and D. Yuret. 2007. The CONLL 2007 shared task on dependency parsing. In *Proc. of the CoNLL 2007 Shared Task Session*, pp. 915–932.

F. Sangati and W. Zuidema. 2008. Unsupervised Methods for Head Assignments. In *Proc. of the EACL 2009 Conference*, pp. 701–709.