Forest Reranking through Subtree Ranking

Richárd Farkas, Helmut Schmid
Institute for Natural Language Processing
University of Stuttgart
{farkas,schmid}@ims.uni-stuttgart.de

Abstract

We propose the *subtree ranking* approach to *parse forest reranking* which is a generalization of current perceptron-based reranking methods. For the training of the reranker, we extract competing local subtrees, hence the training instances (candidate subtree sets) are very similar to those used during beam-search parsing. This leads to better parameter optimization. Another chief advantage of the framework is that arbitrary learning to rank methods can be applied. We evaluated our reranking approach on German and English phrase structure parsing tasks and compared it to various state-of-the-art reranking approaches such as the perceptron-based forest reranker. The subtree ranking approach with a Maximum Entropy model significantly outperformed the other approaches.

1 Introduction

Reranking has become a popular technique for solving various structured prediction tasks, such as phrase-structure (Collins, 2000) and dependency parsing (Hall, 2007), semantic role labeling (Toutanova et al., 2008) and machine translation (Shen et al., 2004). The idea is to (re)rank candidates extracted by a base system exploiting a rich feature set and operating at a global (usually sentence) level. Reranking achieved significant gains over the base system in many tasks because it has access to information/features which are not computable in the base system. Reranking also outperforms discriminative approaches which try to handle the entire candidate universe (cf. Turian et al. (2006)) because the base system effectively and efficiently filters out many bad candidates and makes the problem tractable.

The standard approach for reranking is the n-best list ranking procedure, where the base system extracts its top n global-level candidates with associated goodness scores that define an initial ranking. Then the task is to rerank these candidates by using a rich feature set. The bottleneck of this approach is the small number of candidates considered. Compared to n-best lists, packed parse forests encode more candidates in a compact way. **Forest reranking** methods have been proposed, which can exploit the richer set of candidates and they have been successfully applied for phrase-structure (Huang, 2008), dependency (Hayashi et al., 2011) parsing and machine translation (Li and Khudanpur, 2009) as well.

Huang (2008) introduced the **perceptron-based forest reranking** approach. The core of the algorithm is a beam-search based decoder operating on the packed forest in a bottom-up manner. It follows the assumption that the feature values of the whole structure are the sum of the feature values of the local elements and they are designed to the usage of the perceptron update. Under these assumptions a 1-best Viterbi or beam-search decoder can be efficiently employed at parsing and training time. During training, it decodes the 1-best complete parse then it makes the perceptron update against the oracle parse, i.e. the perceptron is trained at the global (sentence) level.

We propose here a **subtree ranker** approach which can be regarded as a generalization of this for-

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 1038–1047, Jeju Island, Korea, 12–14 July 2012.

est reranking procedure. In contrast to updating on a single (sub)tree per sentence using only the 1-best parse (perceptron-based forest reranking), the subtree ranker exploits subtrees of all sizes from a sentence and trains a (re)ranker utilising several derivations of the constituent in question. During parsing we conduct a beam-search extraction by asking the ranker to select the k best subtrees among the possible candidates of each forest node. The chief motivation for this approach is that in this way, training and prediction are carried out on similar local candidate lists which we expect to be favorable to the learning mechanism. We empirically prove that the trained discriminative rankers benefit from having access to a larger amount of subtree candidates. Moreover, in this framework any kind of learning to rank methods can be chosen as ranker, including pair-wise and list-wise classifiers (Li, 2011).

The contributions of this paper are the following:

- We extend the perceptron-based forest rerankers to the subtree ranker forest reranking framework which allows to replace the perceptron update by any kind of learning to rank procedure.
- We report experimental results on German and English phrase-structure parsing comparing subtree rerankers to various other rerankers showing a significant improvement over the perceptron-based forest reranker approach.

2 Related Work

Our method is closely related to the work of Huang (2008), who introduced forest-based reranking for phrase structure parsing. The proposed framework can be regarded as an extension of this approach. It has several advantages compared with the perceptron-based forest reranker. In this paper we focus on the most important one – and briefly discuss two others in Section 5 – which is enabling the use of any kind of learning to rank approaches. While the perceptron is fast to train, other machine learning approaches usually outperform it. Most of the existing learning to rank approaches are built on linear models and evaluate the candidates independently of each other (such as MaxEnt (Charniak and Johnson, 2005), SVMRank (Joachims, 2002), SoftRank (Guiver and Snelson, 2008)). Thus the choice of the learning method does not influence parsing time. We believe that the real bottleneck of parsing applications is parsing time and not training time. On the other hand, they can learn a better model (at the cost of higher training time) than the Perceptron. In theory, we can imagine learning to rank approaches which can not be reduced to the individual scoring of candidates at prediction time, for instance a decision tree-based pairwise ranker. Although such methods would also fit into the general subtree framework, they are not employed in practice (Li, 2011).

The subtree ranking approach is a generalization of the perceptron-based approach. If the ranking algorithm is the Averaged Perceptron, the parsing algorithm reduces to perceptron-based forest parsing. If the "selection strategy" utilizes the base system ranking and training starts with a filtering step which keeps only candidate sets from the root node of the forest we get the offline version of the training procedure of the perceptron-based forest reranker of Huang (2008).

As our approach is based on local ranking (local update in the online learning literature), it is highly related to early update which looks for the first local decision point where the oracle parse falls out from the beam. Early update was introduced by Collins and Roark (2004) for incremental parsing and adopted to forest reranking by Wang and Zong (2011).

Besides phrase structure parsing, the forest reranking approach was successfully applied for dependency parsing as well. Hayashi et al. (2011) introduced a procedure where the interpolation of a generative and a forest-based discriminative parser is exploited.

From the algorithmic point of view, our approach is probably most closely related to Searn (Daumé et al., 2009) and Magerman (1995) as we also employ a particular machine learned model for a sequence of local decisions. The topological order of the parse forest nodes can form the "sequence of choices" of Searn. The biggest differences between our approach and Searn are that we propose an approach employing beam search and the "policy" is a ranker in our framework instead of a multiclass classifier as there are no "actions" here, instead we have to choose from candidate sets in the forest reranking

framework. In a wider sense, our approach can be regarded – like Searn – as an Inverse Reinforcement Learning approach where "one is given an environment and a set of trajectories and the problem is to find a reward function such that an agent acting optimally with respect to the reward function would follow trajectories that match those in the training set" (Neu and Szepesvári, 2009). Neu and Szepesvári (2009) introduced the top-down parsing Markov Decision Processes and experiment with several inverse reinforcement learning methods. The forest reranking approaches are bottom-up parsers which would require a new (non-straightforward) definition of a corresponding Markov Decision Process.

3 Subtree Ranking-based Forest Reranking

A **packed parse forest** is a compact representation of possible parses for a given sentence. A forest has the structure of a hypergraph, whose nodes V are the elementary units of the underlying structured prediction problem and the hyperedges E are the possible deductive steps from the nodes. In this paper we experimented with phrase-structure parse reranking. In this framework *nodes* correspond to constituents spanning a certain scope of the input sentence and a *hyperedge* e links a parent node $head(e)$ to its children $tails(e)$ (i.e. a hyperedge is a CFG rule in context).

The forest is extracted from the chart of a base PCFG parser, usually employing a heavy pruning strategy. Then the goal of a forest reranker is to find the best parse of the input sentence exploiting a feature representation of (sub)trees.

We sketch the **parsing** procedure of the subtree ranker in Algorithm 1. It is a bottom-up beam-search parser operating on the hypergraph. At each node v we store the k best subtrees $S(v)$ headed by the node. The $S(v)$ lists contain the k top-ranked subtrees by the ranker R among the candidates in the beam. The set of candidate subtrees at a node is the union of the candidates at the different hyperedges. The set of candidate subtrees at a certain hyperedge, in turn, is formed by the Cartesian product $\otimes S(v_i)$ of the k-best subtrees stored at the child nodes v_i. The final output of forest ranking is the 1-best subtree headed by the goal node $S_1(v_{goal})$.

Algorithm 1 Subtree Ranking

Require: $\langle V, E \rangle$ forest, R ranker
  for all $v \in V$ in bottom-up topological order **do**
    $C \leftarrow \emptyset$
    for all $e \in E, head(e) = v$ **do**
      $C \leftarrow C \cup (\otimes S(v_i)), v_i \in tails(e)$
    end for
    $S(v) \leftarrow R_k(C)$
  end for
  return $S_1(v_{goal})$

For **training** the ranker we propose to extract local candidate lists from the forests which share the characteristics of the candidates at parsing time. Algorithm 2 depicts the training procedure of the subtree ranker.

As forests sometimes do not contain the gold standard tree, we extract an oracle tree instead, which is the closest derivable tree in the forest to the gold standard tree (Collins, 2000). Then we optimize the parser for ranking the oracle tree at the top. This procedure is beneficial to training since the objective is a reachable state. In Algorithm 2, we extract the oracle tree from the parses encoded in the forest $\langle V, E \rangle_i$ for the ith training sentence, which is the tree with the highest F-score when compared to the gold standard tree y_i. For each of the training sentences we calculate the oracle subtrees for each node $\{O_v\}$ of the corresponding parse forest. We follow the dynamic programming approach of Huang (2008) for the extraction of the forest oracle. The goal of this algorithm is to extract the full oracle tree, but as a side product it calculates the best possible subtree for all nodes including the nodes outside of the full oracle tree as well.

After computing the oracle subtrees, we crawl the forests bottom-up and extract a training instance $\langle C, O_v \rangle$ at each node v which consists of the candidate set C and the oracle O_v at that node. The creation of candidate lists is exactly the same as it was at parsing time. Then we create training instances from each of the candidate lists and form the set of subtrees $S(v)$ which is stored for candidate extraction at the higher levels of the forest (later steps in the training instance extraction).

A crucial design question is how to form the $S(v)$ sets during training, which is the task of the *selection*

Algorithm 2 Subtree Ranker Training

Require: $\{\langle V, E\rangle_i, y_i\}_1^N$, SS selection strategy
  $T \leftarrow \emptyset$
  for all $i \leftarrow 1...N$ **do**
    $O \leftarrow$ oracle extractor$(\langle V, E\rangle_i, y_i)$
    for all $v \in V_i$ in bottom-up topological order **do**
      $C \leftarrow \emptyset$
      for all $e \in E, head(e) = v$ **do**
        $C \leftarrow C \cup (\otimes S(v_j)), v_j \in tails(e)$
      end for
      $T \leftarrow T \cup \langle C, O_v\rangle$
      $S(v) \leftarrow SS(C, O_v)$
    end for
  end for
  $R \leftarrow$ train reranker(T)
  return R

strategy SS. One possible solution is to keep the k best oracle subtrees, i.e. the k subtrees closest to the gold standard parse, which is analogous to using the gold standard labels in Maximum Entropy Markov Models for sequence labeling problems (we refer this selection strategy as 'oracle subtree' later on). The problem with this solution is that if the rankers have been trained on the oracle subtrees potentially leads to a suboptimal performance as the outputs of the ranker at prediction time are noisy. Note that this approach is not a classical beam-based decoding anymore as the "beam" is maintained according to the oracle parses and there is no model which influences that. An alternative solution – beam-based decoding – is to use a ranker model to extract the $S(v)$ set in training time as well. In the general reranking approach, we assume that the ranking of the base parser is reliable. So we store the k best subtrees according to the base system in $S(v)$ (the 'base system ranking' selection strategy). Note that the general framework keeps this question open and lets the implementations define a selection strategy SS.

After extracting the training instances T we can train an arbitrary ranker R offline. Note that the extraction of candidate lists is exactly the same in Algorithm 1 and 2 while the creation of S_v can be different.

4 Experiments

We carried out experiments on English and German phrase-structure reranking. As evaluation metric, we used the standard `evalb` implementation of PARSEVAL on every sentence without length limitation and we start from raw sentences without gold standard POS tagging. As the grammatical functions of constituents are important from a downstream application point of view – especially in German – we also report PARSEVAL scores on the conflation of constituent labels and grammatical functions. These scores are shown in brackets in Table 2.

4.1 Datasets

We used the Wall Street Journal subcorpus of the Ontonotes v4.0 corpus (Weischedel et al., 2011)[1] for English. As usual sections 2-21, 23 and 24 served as training set (30,060 sentences), test set (1,640 sentences), and development set (1,336 sentences), respectively. Using the Ontonotes version enables us to assess parser robustness. To this end, we evaluated our models also on the weblog subcorpus of the Ontonotes v4.0 corpus which consists of 15,103 sentences.

For German we used the Tiger treebank (Brants et al., 2002). We take the first 40,474 sentences of the Tiger treebank as training data, the next 5,000 sentences as development data, and the last 5,000 sentences as test data.

4.2 Implementation of the Generic Framework

We investigate the Averaged Perceptron and a Maximum Entropy ranker as the reranker R in the subtree ranking framework. The Maximum Entropy ranker model is optimized with a loss function which is the negative log conditional likelihood of the oracle trees relative to the candidate sets. In the case of multiple oracles we optimize for the sum of the oracle trees' posterior probabilities (Charniak and Johnson, 2005).

In our setup the parsing algorithm is identical to the perceptron-based forest reranker of Huang (2008) because both the Averaged Perceptron and the Maximum Entropy rankers score the local subtree candidates independently of each other using

[1] Note that it contains less sentences and a slightly modified annotation schema than the Penn Treebank.

a linear model. There is no need to compute the global normalization constant of the Maximum Entropy model because we only need the ranking and not the probabilities. Hence the difference is in how to train the ranker model.

We experimented with both the 'oracle subtree' and the 'base system ranking' selection strategies (see Section 3).

4.3 Five Methods for Forest-based Reranking

We conducted comparative experiments employing the proposed subtree ranking approach and state-of-the-art methods for forest reranking. Note that they are equivalent in parsing time as each of them uses beam-search with a linear classifier, on the other hand they are radically different in their training.

- The original perceptron-based forest reranker of Huang (2008) ('perceptron with global training').
- The same method employing the early-update updating mechanism instead of the global update. Wang and Zong (2011) reported a significant gain using this update over the standard global update ('perceptron with early update').
- Similar to learning a perceptron at the global level and then applying it at local decisions, we can train a Maximum Entropy ranker at the global level utilizing the n-best full parse candidates of the base parser, then use this model for local decision making. So we train the standard n-best rerankers (Charniak and Johnson, 2005) and then apply them in the beam-search-based Viterbi parser ('n-best list training'). Applying the feature weights adjusted in this approach in the forest-based decoding outperforms the standard n-best list decoding by an F-score of 0.3 on the German dataset.
- The subtree ranker method using the Averaged Perceptron reranker. This is different from the 'perceptron with global training' as we conduct updates at every local decision point and we do offline training ('subtree ranking by AvgPer').
- The subtree ranker method using Maximum Entropy training ('subtree ranking by MaxEnt').

We (re)implemented these methods and used the same forests and the same feature sets for the comparative experiments.

4.4 Implementation Details

We used the first-stage **PCFG parser** of Charniak and Johnson (2005) for English and BitPar (Schmid, 2004) for German. BitPar employs a grammar engineered for German (for details please refer to Farkas et al. (2011)). These two parsers are state-of-the-art PCFG parsers for English and German, respectively. For German the base parser and the reranker operate on the conflation of constituent labels and grammatical functions. For English, we used the forest extraction and pruning code of Huang (2008). The pruning removes hyperedges where the difference between the cost of the best derivation using this hyperedge and the cost of the globally best derivation is above some threshold. For German, we used the pruned parse forest of Bitpar (Schmid, 2004). After computing the posterior probability of each hyperedge given the input sentence, Bitpar prunes the parse forest by deleting hyperedges whose posterior probability is below some threshold. (We used the threshold 0.01).

We employed an Averaged Perceptron (for 'perceptron with global training', 'perceptron with early update' and 'subtree ranking by AvgPer') and a Maximum Entropy **reranker** (for 'subtree ranking by MaxEnt' and 'n-best list training'). For the perceptron reranker, we used the Joshua implementation[2]. The optimal number of iterations was determined on the development set. For the Maximum Entropy reranker we used the `RankMaxEnt` implementation of the Mallet package (McCallum, 2002) modified to use the objective function of Charniak and Johnson (2005) and we optimized the L2 regularizer coefficient on the development set.

The beam-size were set to 15 (the value suggested by Huang (2008)) during parsing and the training of the 'perceptron with global training' and 'perceptron with early update' models. We used $k = 3$ for training the 'subtree ranking by AvgPer' and 'subtree ranking by MaxEnt' rankers (see Section 5 for a discussion on this).

In the English experiments, we followed (Huang,

[2] `http://joshua.sourceforge.net/Joshua/`

	Tiger test	WSJ dev	WSJ test	WB
base system (1-best)	76.84 (65.91)	89.29	88.63	81.86
oracle tree	90.66 (80.38)	97.31	97.30	94.18

Table 1: The lower and upper bounds for rerankers on the four evaluation datasets. The numbers in brackets refers to evaluation with grammatical function labels on the German dataset.

	Tiger test	WSJ dev	WSJ test	WB
perceptron with global training	78.39 (67.79)	90.58	89.60	82.87
perceptron with early update	78.83 (68.05)	90.81†	90.01	83.03†
n-best list training	78.75 (68.04)	90.89	90.11	83.55
subtree ranking by AvgPer	78.54† (67.97†)	90.65†	89.97	83.04†
subtree ranking by MaxEnt	79.36 (68.72)	91.14	90.32	83.83

Table 2: The results achieved by various forest rerankers. The difference between the scores marked by † and the 'perceptron with global training' were not statistically significant with $p < 0.005$ according to the the McNemar test. All other results are statistically different from this baseline.

2008) and selectively re-implemented **feature templates** from (Collins, 2000) and Charniak and Johnson (2005). For German we re-implemented the feature templates of Versley and Rehbein (2009) which is the state-of-the-art feature set for German. It consists of features constructed from the lexicalized parse tree and its typed dependencies along with features based on external statistical information (such as the clustering of unknown words according to their context of occurrence and PP attachment statistics gathered from the automatically POS tagged DE-WaC corpus, a 1.7G words sample of the German-language WWW). We filtered out rare features which occurred in less than 10 forests (we used the same non-tuned threshold for the English and German training sets as well).

We also re-implemented the oracle extraction procedure of Huang (2008) and extended its convolution and translation operators for using the base system score as tie breaker.

4.5 Results

Table 1 shows the results of the 1-best parse of the base system and the oracle scores – i.e. the **lower and upper bounds** for the rerankers – for the four evaluation datasets used in our experiments. The German and the weblog datasets are more difficult for the parsers.

The following table summarizes the characteristics of the subtree ranker's training sample of the German and English datasets by employing the 'oracle subtree' selection strategy:

	Tiger train	WSJ train
#candidate lists	266,808	1,431,058
avg. size of cand. lists	3.2	5.7
#features before filtering	2,683,552	22,164,931
#features after filtering	94,164	858,610

Table 3: The sizes of the subtree ranker training datasets at $k = 3$.

Using this selection strategy the training dataset is smaller than the training dataset of the n-best list rankers – where offline trainers are employed as well – as the total number of candidates is similar (and even less in the Tiger corpus) while there are fewer firing features at the subtrees than at full trees.

Table 2 summarizes the results achieved by various forest rerankers. Both subtree rankers used the oracle subtrees as the selection strategy of Algorithm 2. The 'subtree ranking by MaxEnt' method significantly outperformed the perceptron-based forest reranking algorithms at each of the datasets and seems to be more robust as its advantage on the out-domain data 'WB' is higher compared with the in-domain 'WSJ' datasets. The early update improves the perceptron based forest rerankers which is in line with the results reported by Wang and Zong (2011). The 'n-best list training' method works surprisingly well. It outperforms both perceptron-based forest

rerankers on the English datasets (while achieving a smaller F-score than the perceptron with early update on the Tiger corpus) which demonstrates the potential of utilizing larger candidate lists for discriminative training of rerankers. The comparison of the 'subtree ranking by AvgPer' row and the 'subtree ranking by MaxEnt' row shows a clear advantage of the Maximum Entropy training mechanism over the Averaged Perceptron.

Besides the 'oracle subtree' **selection strategy** we also experimented with the 'base system ranking' selection strategy with subtree Maximum Entropy ranker. Table 4 compares the accuracies of the two strategies. The difference between the two strategies varies among datasets. In the German dataset, they are competitive and the prediction of grammatical functions benefits from the 'base system ranking' strategy, while it performs considerably worse at the English datasets.

	Tiger test	WSJ test	WB
oracle SS	79.36 (68.72)	90.32	83.83
base sys SS	79.34 (68.84)	89.97	83.34

Table 4: The results of the two selection strategies. Using the oracle trees proved to be better on each of the datasets.

Extracting candidate lists from each of the local decision points might seem to be redundant. To gain some insight into this question, we investigated the effect of training instance filtering strategies on the Tiger treebank. We removed the training instances from the training sample T where the F-score of the oracle (sub)tree against the gold standard tree is less than a certain threshold (this data selection procedure was inspired by Li and Khudanpur (2008)). The idea behind this **data selection** is to eliminate bad training examples which might push the learner into the wrong direction. Figure 1 depicts the results on the Tiger treebank as a function of this data selection threshold.

With this data selection strategy we could further gain 0.22 F-score percentage points achieving 79.58 (68.87) and we can conclude that omitting candidate sets far from the gold-standard tree helps training. Figure 1 also shows that too strict filtering hurts the performance. The result with threshold=90 is worse than the result without filtering. We should note that similar data selection methods can be applied

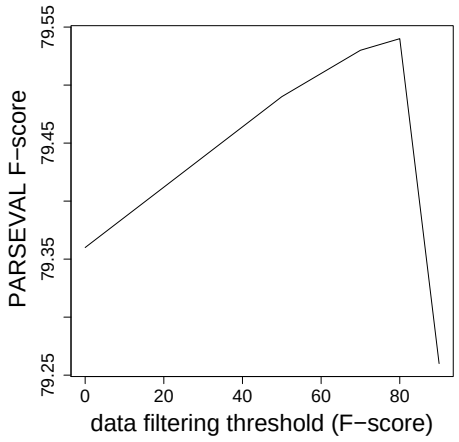

Figure 1: The effect of data selection on the Tiger test set.

to each of the baseline systems and the comparison to them would be fair with conducting that. Thus we consider our results without data selection to be final.

5 Discussion

We experimentally showed in the previous section that the subtree forest reranking approach with Maximum Entropy models significantly outperforms the perceptron-based forest reranking approach. This improvement must be the result of differences in the training algorithms because there is no difference between the two approaches at parse time, as we discussed in Section 4.2.

There are two sources of these improvements. (i) We use local subtrees as training instances instead of using the global parses exclusively. The most important difference between the training of the perceptron-based forest reranker and the subtree forest reranker is that we train on subtrees (extract candidate sets) outside of the Viterbi parses as well, i.e. our intuition is that the training of the discriminative model can benefit from seeing good and bad subtrees far from the best parses as well. (ii) The subtree ranker framework enables us to employ the Maximum Entropy ranker on multiple candidates, which usually outperforms the Averaged Perceptron.

The results of Table 2 can be considered as two paths from the 'perceptron with global training' to the 'subtree ranking by MaxEnt' applying these

sources of improvements. If we use (i) and stay with the Averaged Perceptron as learning algorithm we get 'subtree ranking by AvgPer'. If we additionally replace the Averaged Perceptron by Maximum Entropy – i.e. follow (ii) – we arrive at 'subtree ranking by MaxEnt'. On the other hand, the 'n-best training' uses global trees and Maximum Entropy for training, so the reason of the difference between 'perceptron with global training' and 'n-best training' is (ii). Then we arrive at 'subtree ranking by MaxEnt' by (i). This line of thoughts and the figures of Table 2 indicate that the added value of (i) and (ii) are similar in magnitude.

5.1 Error Analysis

For understanding the added value of the proposed subtree ranking method, we manually investigated sentences from the German development set and compared the parses of the 'perceptron with global training' with the 'subtree ranking by MaxEnt'. We could not found any linguistic phenomena which was handled clearly better by the subtree ranker[3], but it made considerably more fixes than errors in the following cases:

- the attachment of adverbs,
- the unary branching verbal phrases and
- extremely short sentences which does not contain any verb (fragments).

5.2 Novel Opportunities with the Subtree Ranking Framework

A generalization issue of the subtree ranking approach is that it allows to use **any kind of feature representation and arbitrary aggregation** of local features. The basic assumption of training on the global (sentence) level in the perceptron reranking framework is that the feature vector of a subtree is the sum of the feature vectors of the children and the features extracted from the root of the subtree in question. This decomposability assumption provides a fine framework in the case of binary features which fire if a certain linguistic phenomenon occurs. On the other hand, this is not straightforward in the presence of real valued features. For example, Versley and Rehbein (2009) introduce real-valued features for supporting German PP-attachment recognition – the mutual information of noun and preposition co-occurrence estimated from a huge unlabeled corpus – and this single feature template (about 80 features) could achieve a gain of 1 point in phrase structure parsing accuracy while the same improvement can be achieved by several feature templates and millions of binary features. The aggregation of such feature values can be different from summing, for instance the semantics of the feature can demand averaging, minimum, maximum or introducing new features etc. Another opportunity for extending current approaches is to employ utility functions on top of the sum of the binary feature values. Each of these extensions fits into the proposed framework.

The subtree ranking framework also enables the usage of **different models at different kinds of nodes**. For example, different models can be trained for ranking subtress headed by noun phrases and for verb phrases. This is not feasible in the perceptron-based forest ranker which sums up features and updates feature weights at the sentence level while the ranker R in Algorithm 2 can refer to several models because we handle local decisions separately. This approach would not hurt parsing speed as one particular model is asked at each node, but it multiplies memory requirements. This is an approach which the subtree ranking framework allows, but which would not fit to the global level updates of the perceptron forest rerankers.

As a first step in this direction of research we experimented with training three different Maximum Entropy models using the same feature representation, the first only on candidate lists extracted from noun phrase nodes, the second on verb phrase nodes and the third on all nodes (i.e. the third model is equivalent to the 'subtree MaxEnt' model). Then at prediction time, we ask that model (out of the three) which is responsible for ranking the candidates of the current type of node. This approach performed worse than the single model approach achieving an F-scores of 79.24 (68.46) on the Tiger test dataset. This negative results – compared with 79.36 (68.72) achieved by a single model – is probably due to data sparsity problems. The amount of training samples for noun phrases is 6% of the full training sample

[3] We believe that this might be the case only if we would introduce new information (e.g. features) for the system.

and it seems that a better model can be learned from a much bigger but more heterogeneous dataset.

5.3 On the Efficiency of Subtree Ranking

In subtree ranking, we extract a larger number of training instances (candidate lists) than the perceptron-based approach which extracts exactly one instance from a sentence. Moreover, the candidate lists are longer than the perceptron-based approach (where 2 "candidates" are compared against each other). Training on this larger set (refer Table 3 for concrete figures) consumes more space and time.

In our implementation, we keep the whole training dataset in the memory. With this implementation the whole training process (feature extraction, candidate extraction and training the Maximum Entropy ranker) takes 3 hours and uses 10GB of memory at $k = 1$ and it takes 20 hours and uses 60GB of memory at $k = 3$ ((Huang, 2008) reported 5.3 and 27.3 hours at beam-sizes of 1 and 15 respectively but it used only 1.2GB of memory). The in-depth investigation of the effect of k is among our future plans.

6 Conclusions

We presented a subtree ranking approach to parse forest reranking, which is a generalization of current reranking methods. The main advantages of our approach are: (i) The candidate lists used during training are very similar to those used during parsing, which leads to better parameter optimization. (ii) Arbitrary ranking methods can be applied in our approach. (iii) The reranking models need not to be decomposable.

We evaluated our parse reranking approach on German and English phrase structure parsing tasks and compared it to various state-of-the-art reranking approaches such as the perceptron-based forest reranker (Huang, 2008). The subtree reranking approach with a Maximum Entropy model significantly outperformed the other approaches.

We conjecture two reasons for this result: (i) By training on all subtrees instead of Viterbi parses or n-best parses only, we use the available training data more effectively. (ii) The subtree ranker framework allows us to use a standard Maximum Entropy learner in parse-forest training instead of the Perceptron, which is usually superior.

Acknowledgements

We thank Liang Huang to provide us the modified version of the Charniak parser, which output a packed forest for each sentence along with his forest pruning code.

This work was founded by the Deutsche Forschungsgemeinschaft grant SFB 732, project D4.

References

Sabine Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The TIGER treebank. In *Proceedings of the Workshop on Treebanks and Linguistic Theories*, pages 24–41.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180.

Michael Collins and Brian Roark. 2004. Incremental parsing with the perceptron algorithm. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 111–118.

Michael Collins. 2000. Discriminative reranking for natural language parsing. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 175–182.

Hal Daumé, III, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine Learning*, 75(3):297–325, June.

Richàrd Farkas, Bernd Bohnet, and Helmut Schmid. 2011. Features for phrase-structure reranking from dependency parses. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 209–214.

John Guiver and Edward Snelson. 2008. Learning to rank with softrank and gaussian processes. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 259–266.

Keith Hall. 2007. K-best spanning tree parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 392–399, June.

Katsuhiko Hayashi, Taro Watanabe, Masayuki Asahara, and Yuji Matsumoto. 2011. Third-order variational reranking on packed-shared dependency forests. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1479–1488.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Proceedings of ACL-08: HLT*, pages 586–594.

T. Joachims. 2002. Optimizing search engines using clickthrough data. In *ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142.

Zhifei Li and Sanjeev Khudanpur. 2008. Large-scale discriminative n-gram language models for statistical machine translation. In *Proceedings of the 8th AMTA conference*, pages 133–142.

Z. Li and S. Khudanpur, 2009. *GALE book chapter on "MT From Text"*, chapter Forest reranking for machine translation with the perceptron algorithm.

Hang Li. 2011. *Learning to Rank for Information Retrieval and Natural Language Processing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

David M. Magerman. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, June.

Andrew Kachites McCallum. 2002. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu.

Gergely Neu and Csaba Szepesvári. 2009. Training parsers by inverse reinforcement learning. *Machine Learning*, 77(2–3):303–337.

Helmut Schmid. 2004. Efficient parsing of highly ambiguous context-free grammars with bit vectors. In *Proceedings of Coling 2004*, pages 162–168.

Libin Shen, Anoop Sarkar, and Franz Josef Och. 2004. Discriminative reranking for machine translation. In Daniel Marcu Susan Dumais and Salim Roukos, editors, *HLT-NAACL 2004: Main Proceedings*, pages 177–184.

Kristina Toutanova, Aria Haghighi, and Christopher D. Manning. 2008. A global joint model for semantic role labeling. *Computational Linguistics*, 34(2):161–191.

Joseph P. Turian, Benjamin Wellington, and I. Dan Melamed. 2006. Scalable discriminative learning for natural language parsing and translation. In *NIPS*, pages 1409–1416.

Yannick Versley and Ines Rehbein. 2009. Scalable discriminative parsing for german. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 134–137.

Zhiguo Wang and Chengqing Zong. 2011. Parse reranking based on higher-order lexical dependencies. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 1251–1259.

Ralph Weischedel, Eduard Hovy, Martha Palmer, Mitch Marcus, Robert Belvin, Sameer Pradhan, Lance Ramshaw, and Nianwen Xue, 2011. *Handbook of Natural Language Processing and Machine Translation.*, chapter OntoNotes: A Large Training Corpus for Enhanced Processing.