END-TO-END LEARNING OF PARSING MODELS FOR INFORMATION RETRIEVAL

Jennifer Gillenwater[*]*, Xiaodong He, Jianfeng Gao, Li Deng*

jengi@seas.upenn.edu, {xiaohe,jfgao,deng}@microsoft.com

Microsoft Research, One Microsoft Way, Redmond, WA 98052, USA

ABSTRACT

Parsers have been shown to be helpful in information retrieval tasks because they are able to model long-span word dependencies efficiently. While previous work focused on using traditional syntactic parse trees, this paper proposes a new approach where, unlike previous work, the parser parameters are discriminatively trained to directly optimize a non-convex and non-smooth IR measure. The relevance between a document and a query is then modeled by the weighted tree edit distance between their parses. We evaluate our method on a large scale web search task consisting of a real world query set. Results show that the new parser is more effective for document retrieval than using traditional syntactic parse trees. It gives significant improvement, especially for long queries where proper modeling of long-span dependencies is crucial.

Index Terms— information retrieval, parsing model, end-to-end optimization, tree edit distance

1. INTRODUCTION

A long query can often better express a user's intent than a short query. However, search results for long queries are notoriously worse than those for short queries, e.g., the poor performance of search engines for queries with five or more words is well-documented in [2]. In the current work, we tackle this problem using dependency parsers. Dependency parsing models have been shown to be helpful in information retrieval (IR) tasks because they are an efficient means for exploiting longer-span word dependencies than just those within a noun phrase or between adjacent words. Previous work in the area of parsing models for IR includes [18][11][20]. Table 1 summarizes two key differences between such earlier methods and the work to be presented in this paper.

First, our ranking function, weighted tree edit distance (TED), is novel. Unlike earlier rankers that compute the likelihood of generating a document from a query or vice versa, we are not constrained to probability space. Further, unlike un-weighted TED functions that simply assign a constant cost for each type of tree edit operation (see Section 3), we condition on the characteristics of the tree nodes involved when deciding on the cost and use this as the basis for parser optimization. Both of these differences make our ranker more flexible and easier to optimize for IR.

The second important contribution of this work is the automatic learning of the parser parameters with the goal of directly optimizing the end-to-end IR measure — mean Normalized Discounted Cumulative Gain (NDCG) [15]. Each query-document pair in our dataset has a human-annotated relevance label that is an integer between 0 (document being irrelevant) and 4 (document being very relevant). This serves as our source of supervision. The goal is to train the parser such that TED correlates with relevance. Previous methods have either: 1) learned the parser parameters in an unsupervised manner, which fails to take advantage of the supervision information available from relevance judgments, or 2) learned the parser parameters in a supervised manner but from a supervision source that fails to match the document retrieval task, such as the standard syntactic parses of the Wall Street Journal. The method we propose here not only is supervised but also relies on a supervision source that is well-matched to the IR task.

Table 1. Summary of previous work

	Ranking	Parameter Optimization
Nallapati and Allan [18]	Likelihood	unsupervised: word co-occurrence counts
Gao et al. [11]	Likelihood	unsupervised: Viterbi EM to optimize likelihood
Punuakanok et al. [20]	unweighted TED	supervised: standard syntactic trees [7]
This work	weighted TED	supervised: optimize NDCG

However, training parsers using IR measures is difficult in general. Typical IR measures [22], viewed as functions of the ranker scores, are either flat or discontinuous

[*] Work performed while an intern at Microsoft Research.

everywhere [4]. Additionally, the measures require sorting by score, which itself is a non-differentiable operation. The NDCG relevance measure we use is no exception. Formally, for a given query **q**, NDCG is defined as:

$$\text{NDCG@L} = \frac{1}{Z}\sum_{i=1}^{L}\frac{2^{v_i}-1}{log_2(1+i)} \tag{1}$$

where $v_i \in \{0, \ldots, 4\}$ is the label for the relevance level of the *i*-th document to **q** in the sorted list and Z is a normalization constant computed such that NDCG@L = 1 for a perfect ranking of the top L documents. For multiple queries, the NDCGs are simply averaged. This measure expresses the key intuition that the higher a relevant document appears on a list of search results, the better.

It is easy to verify that NDCG, if used as an objective function, is non-smooth, and thus presents a particular challenge to most optimization approaches that require gradient computation. RankNet [5] solves this problem by using an objective whose gradient can be easily computed but whose value is only loosely coupled with NDCG. LambdaRank [6], an improved version of RankNet, amounts to scaling the gradients of RankNet by a function of NDCG. In this work, our goal is to optimize parser parameters to maximize NDCG; that is, to ensure that the TED between the parse of a query and the parse of a relevant document's title is small. Thus, we use the LambdaRank objective to optimize the parser parameters. In short, we do so by defining our ranker to be a function of the parser parameters, which enables us to take gradients of the LambdaRank objective with respect to these parameters.

2. DEPENDENCY PARSING MODEL

The parsing model we use employs independent, directed links. Given a sequence of words $\mathbf{w} = w_1 \ldots w_n$, let $T_\mathbf{w}$ refer to any projective dependency tree for this sequence. Our model assigns the following probability to the parse:

$$p_\theta\,(T_\mathbf{w}) = \prod_{w_i \to w_j \in T_\mathbf{w}} \theta_{w_j|w_i} \tag{2}$$

where $w_i \to w_j$ denotes w_i is the parent of w_j, and $\theta_{w_j|w_i} = p(w_i \to w_j)$. We use $\theta = \{\theta_{w_j|w_i}\}$ to denote the entire set of parser parameters. In practice, to combat sparsity problems, instead of having parameters for each pair of words, we group words into semantically meaningful categories by hierarchical word clustering [12] and have parsing parameters for each pair of categories. In our experiments, 32 clusters are created by building a binary word clustering tree with 6 levels. Additionally, note that we only parse document titles, as the title is the most effective portion of a document for web document retrieval [10].

3. WEIGHTED TED RANKER

To quantify the relevance of a particular document **d** to a query **q**, we assign each (**q**, **d**) pair a score based on the weighted edit distance between their parse trees. Formally, let N(T) denote the set of nodes in parse tree T, and let M represent the set of node substitutions: $M = \{(i,j): i \mapsto j, \mathbf{q}_i \in N(T_\mathbf{q}), \mathbf{d}_j \in N(T_\mathbf{d})\}$, where $T_\mathbf{q}$ and $T_\mathbf{d}$ denote the query tree and the document tree, respectively, $\mathbf{q}_i$ and $\mathbf{d}_j$ denote the i-th and the j-th node in $T_\mathbf{q}$ and $T_\mathbf{d}$, respectively, and $i \mapsto j$ indicates $\mathbf{d}_j$ substitutes for $\mathbf{q}_i$. Similarly, let ε denote an empty node, and define J as the insertion set: $J = \{j: \varepsilon \mapsto j, \mathbf{d}_j \in N(T_\mathbf{d})\}$, and I as the deletion set: $I = \{i: i \mapsto \varepsilon, \mathbf{q}_i \in N(T_\mathbf{q})\}$. Then the TED scoring function is:

$$f(\mathbf{q},\mathbf{d}) = \sum_{(i,j)\in M} g_M(x_i, y_j) + \sum_{j\in J} g_J(y_j) + \sum_{i\in I} g_I(x_i) \tag{3}$$

where we define x_i as shorthand for the parameter associated with the creation of node i in the query tree $T_\mathbf{q}$ (i.e. $\theta_{w_i|\text{Pa}(i)}$), and y_j analogously. We use the algorithm of [9] for computing the sets M, J, I that give the minimum TED value. For the cost functions $g(\cdot)$ we experimented with a few variations. The functions we found to work best take the following forms:

$$g_M(x_i, y_j) = \mathbf{1}(\mathbf{q}_i \neq \mathbf{d}_j \vee \mathbf{q}_{\text{Pa}(i)} \neq \mathbf{d}_{\text{Pa}(j)})\frac{(x_i + y_j)}{\ln(l)} \tag{4}$$
$$g_J(y_j) = 0,\; g_I(x_i) = x_i$$

For substitution, the cost g_M is zero if a "match condition" is satisfied, i.e., the cost is zero if both the nodes involved in the substitution match (are in the same cluster at the 6th level of the tree built using [12]) and their parents match. Otherwise, the substitution cost is a sum of parser parameters. To provide finer granularity, the cost is further scaled by $\frac{1}{\ln(l)}$, where l is a measure of how related the words at the nodes corresponding to x_i and y_i are. Specifically, we check the match condition at each level of the clustering tree from level 6 up until it is satisfied. We then set l equal to the satisfying level #, plus an offset of 2 to ensure $\frac{1}{\ln(l)} < 1$.

The insertion and deletion costs are simpler. For insertion, the cost is zero since a document title is often longer than a query even if the document is very relevant. For deletion, the cost function corresponds to paying a cost proportional to the certainty of the corresponding branch in the parse tree.

4. TRAINING PARSING MODEL FOR NDCG

We now define an objective function for optimizing the parser parameters θ. The design of the objective follows the pairwise learning-to-rank paradigm outlined in [5][6]. Consider a query $\mathbf{q}^{(k)}$ and two documents, $\mathbf{d}^{(h)}$ and $\mathbf{d}^{(s)}$, and suppose $\mathbf{d}^{(h)}$ is more relevant to the query than $\mathbf{d}^{(s)}$. We define the discriminant function:

$$d_{k,h,s} = f\left(\mathbf{q}^{(k)}, \mathbf{d}^{(h)}\right) - f\left(\mathbf{q}^{(k)}, \mathbf{d}^{(s)}\right) \qquad (5)$$

Intuitively, we want to learn a model to increase $d_{k,h,s}$. Thus, we use the following logistic loss over $d_{k,h,s}$, which can be shown to upper bound the pairwise accuracy:

$$C_{k,h,s} = \log(1 + e^{-d_{k,h,s}}) \qquad (6)$$

Note that $C_{k,h,s}$ is convex in $d_{k,h,s}$. The overall objective is expressed in terms of this cost function as:

$$\min_{\theta} \sum_{k=1}^{|Q|} \sum_{h=1}^{|D_{\mathbf{q}^{(k)}}|} \sum_{s=h+1}^{|D_{\mathbf{q}^{(k)}}|} C_{k,h,s} \qquad (7)$$

where Q is the set of all queries and $D_{\mathbf{q}^{(k)}}$ is the set of documents for query $\mathbf{q}^{(k)}$, sorted by relevance judgment. To ensure normalization and non-negativity, we add to the objective the following constraints:

$$\sum_{w \in V} \theta_{w|u} = 1 \ \forall u \in V, and\ \theta \geqslant 0 \qquad (8)$$

where V is the set of word clusters.

To optimize the objective, we form its Lagrangian dual with Lagrange multipliers λ, v:

$$\max_{v \geq 0, \lambda} \min_{\theta} \sum_{k,h,s} C_{k,h,s} + \sum_{u \in V} \lambda_u \left(\sum_{w \in V} \theta_{w|u} - 1 \right) - \sum_{i=1}^{|\theta|} v_i \theta_i$$

and perform gradient descent on this. The step size for gradient descent is selected using line search [19].

This objective correlates with NDCG, and the correlation can be further improved by scaling parameter updates by the NDCG gain of swapping two documents, as in [6]. i.e., scaling $\frac{\partial C_{k,h,s}}{\partial f(\mathbf{q}^{(k)}, \mathbf{d}^{(s)})}$ by:

$$\gamma = \frac{(2^{v_{r_{k,h}}} - 2^{v_{r_{k,s}}})}{Z} \left(\frac{1}{\log(1 + r_{k,h})} - \frac{1}{\log(1 + r_{k,s})} \right)$$

and scaling $\frac{\partial C_{k,h,s}}{\partial f(\mathbf{q}^{(k)}, \mathbf{d}^{(h)})}$ by $-\gamma$. The $r_{k,h}$ and $r_{k,s}$ in this formula represent the ranks of documents h and s for query k, and v is the relevance label as defined in eq. (1). Observe that this scaling ensures $-\gamma \frac{\partial C_{k,h,s}}{\partial f(\mathbf{q}^{(k)}, \mathbf{d}^{(h)})}$ will be positive, forcing the resulting parameter update to increase $f\left(\mathbf{q}^{(k)}, \mathbf{d}^{(h)}\right)$. [6] showed that following these scaled gradients is equivalent to optimizing an implicit convex objective and so should converge to the objective's global minimum.

A summary of our training procedure is given by Algorithm 1. Note that while the objective function is convex, the overall process is not guaranteed to find a global optimum because parse trees change as the parser parameters are updated (step 3). Thus, TEDs can depend on different parameters from one iteration to the next. In practice we found that, despite the non-convexity of the overall problem, the objective still tends to decrease over time, and training converges quickly after about 20 iterations (Fig. 1).

```
1 Initialize $\theta$ randomly
2 while objective gradient is significant do
3     Parse each $\mathbf{w} \in Q \cup D$: $\arg\max_{T_\mathbf{w}} p_\theta(T_\mathbf{w})$
4     foreach $\mathbf{q} \in Q, \mathbf{d} \in D_\mathbf{q}$ do
5         Compute tree edit distance
6     end
7     Update $\theta$ according to $\gamma$-scaled gradients
8 end
```

Algorithm 1: Training Procedure

5. EXPERIMENTS

We evaluate the retrieval models on a dataset that contains 2,050 English queries, each of which is at least 5 words long, sampled from one year's worth of query logs of a commercial search engine. On average, each query is associated with 185 web documents. In our experiments, the dataset is split into two sets: a training set that contains 80% of the queries and a test set that contains the remaining 20%.

To study the effectiveness of our optimization method for parser parameters, we plot the objective value w.r.t. training iterations. After each update, the objective is always lower than before, since it is convex. However, because we then re-compute the structure of the 1-best parse trees (line 3 in Algorithm 1), the value of the objective tends to increase somewhat before the next parameter update. Nevertheless, overall the objective decreases as desired; the "before" curve that reflects the true objective shows a substantial and relatively smooth decrease.

In the evaluation, we compare the proposed end-to-end (E2E) learning procedure to a maximum likelihood (ML) trained baseline. That is, instead of directly optimizing NDCG, the baseline uses the Viterbi Expectation-Maximization (EM) algorithm to maximize the likelihood of the parse trees. Then the same tree edit distance ranking function is applied to both sets of parse trees.

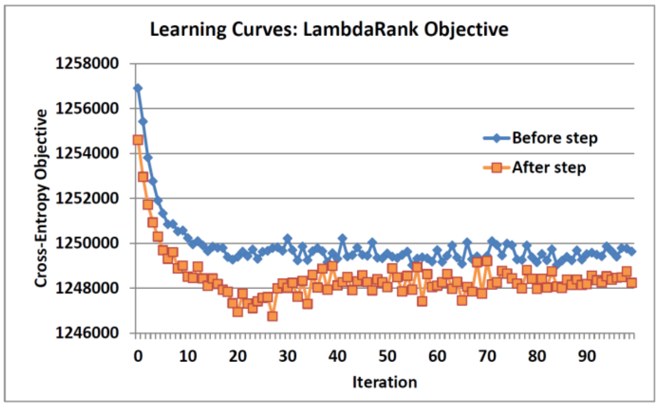

Figure 1: Objective value just before updating the parameters (before line 7 in Algorithm 1) and after updating.

One goal of this work is to better exploit long-span word dependency information to help the IR performance for long queries. In order to study the impact of the proposed method on queries with different lengths, we break down the test set into four groups by query length, and report the results for each group separately, as a percentage of NDCG@10. We also perform a significance test using the paired *t*-test. Differences are considered statistically significant when *p*-value is less than 0.05. Results are summarized in Table 2.

Table 2. Information retrieval results on the test set as a percentage of NDCG@10.

Query length	Number of queries	ML trained	E2E trained	Improve -ment
5	211	32.16	32.27	+0.11
6	92	30.05	30.33	+0.28
7	51	27.69	28.20	$+0.51^{\dagger}$
≥ 8	56	24.52	25.18	$+0.66^{\dagger}$

The superscript † indicates that the improvement is statistically significant ($p < 0.05$).

As shown in Table 2, the end-to-end optimized parsing model outperforms the ML-trained parsing model significantly for queries that contain seven words or more. These results demonstrate that the end-to-end optimized parsing model can better model the long-span word dependency information than the baseline parsing model.

6. OTHER PRIOR WORK

The idea of training the parser to directly optimize the quality of document retrieval traces back to minimum classification error (MCE) training [16][13], and is also similar to the end-to-end decision-feedback training approaches that have been recently applied to speech translation [25] and spoken language understanding [24]. In this work, as shown in the experimental results, we successfully applied this idea to learning parser parameters for IR tasks.

With regard to improving search results for long queries, there are other approaches besides using parsing models. These range from random walks on word graphs [8], to language models [1] and phrase-based translation models [21], to Markov random fields that tie adjacent query words or tie all words within each noun phrase [17]. In contrast to these works, in this paper we tackle the long-query problem using parsing models, for three reasons. First of all, parsing allows us to exploit longer-range dependencies than just those within a noun phrase or between adjacent words. Secondly, by imposing standard parsing constraints requiring that the dependencies in each parse form a projective tree, we can take advantage of existing dynamic programming algorithms for parsing. Lastly, with parse trees we are able to explore a different sort of ranking function than is usually used in IR: tree edit distance.

7. CONCLUSION

We presented a novel method for training a parser for IR. By combining a LambdaRank-based objective with a new weighted TED ranker whose ranks are a function of the parser parameters, we introduced a method for optimizing parser parameters directly for NDCG. Experiments demonstrate that the new training method converges well. Test results show the superiority of this training method over conventional maximum likelihood training.

We could further improve the approach in various ways. Possible avenues of exploration for enhancing this gain include: 1) using a TED that allows for additional operations such as node re-ordering, 2) increasing training set size dramatically by using click data to provide implied relevance judgments, as in [3], 3) learning one parser for queries and a separate parser for document titles, and 4) improving the optimization using methods such as extended Baum-Welch, as was done in [14] for large-scale parallelized discriminative training.

In a different vein, we also intend to pursue optimizing the parser's structure. The current parser design focuses on learning parser parameters only. In future work we hope to also optimize the parser structure by incorporating structured learning techniques published in the recent literature [23].

8. REFERENCES

[1] M. Bendersky and B. Croft. "Discovering key concepts in verbose queries." In Proc. SIGIR, 2008.
[2] M. Bendersky and B. Croft. "Analysis of long queries in a large scale search log." In Proc. WSCD, 2009.

[3] J. Boyan, D. Freitag, and T. Joachims. “A machine learning architecture for optimizing web search engines.” In Proc. AAAI Workshop, 1996.

[4] C. Burges. “Ranking as learning structured outputs.” In Proc. NIPS, 2005.

[5] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. “Learning to rank using gradient descent.” In Proc. ICML, 2005.

[6] C. Burges, R. Rango, and Q. V. Le. “Learning to rank with non-smooth cost functions.” In Proc. NIPS, 2006.

[7] M. Collins. “Three generative, lexicalized models for statistical parsing.” In Proc. ACL, 1997.

[8] K. Collins-Thompson and J. Callan. “Query expansion using random walk models.” In Proc. CIKM 2005.

[9] E. Demaine, S. Mozes, B. Rossman, and O. Weimann. “An optimal decomposition algorithm for tree edit distance.” Transactions on Algorithms, 2009.

[10] J. Gao, X. He, and J.-Y. Nie. “Clickthrough-based translation models for web search: From word models to phrase models.” In Proc. CIKM, 2010.

[11] J. Gao, J.-Y. Nie, G. Wu, and G. Cao. “Dependence language model for in-formation retrieval.” In Proc. SIGIR, 2004.

[12] J. Goodman. JCLUSTER. Toolkit, http://research.microsoft.com/~joshuago, 2002.

[13] X. He, L. Deng, and W. Chou. “A novel learning method for hidden Markov models in speech and audio processing.” In Proc. IEEE Workshop on Multimedia Signal Processing, 2006.

[14] X. He, L. Deng, and W. Chou. “Discriminative learning in sequential pattern recognition.” In IEEE Signal Processing Magazine, September, 2008.

[15] K. Jarvelin and J. Kekalainen. “IR evaluation methods for retrieving highly relevant documents.” In Proc. SIGIR, 2000.

[16] B-H. Juang and S. Katagiri, “Discriminative learning for minimum error classification.” In IEEE Transactions on Signal Processing, 1992.

[17] D. Metzler and B. Croft. “Latent concept expansion using Markov random fields.” In Proc. SIGIR, 2007.

[18] R. Nallapati and J. Allan. “Capturing term dependencies using a language model based on sentence trees.” In Proc. CIKM, 2002.

[19] J. Nocedal and S. Wright. Numerical Optimization, chapter 3. Springer Verlag, 1999.

[20] V. Punuakanok, D. Roth, and W.-T. Yih. “Natural language inference via dependency tree mapping: An application to question answering.” Computational Linguistics, 2004.

[21] S. Reizler, A. Vasserman, I. Tsochantaridis, and Y. Liu. “Statistical machine translation for query expansion in answer retrieval.” In Proc. ACL, 2007.

[22] S. Robertson and H. Zaragoza. “On rank based effectiveness measures and optimization.” In Information Retrieval, 2007.

[23] R. Socher, C. Lin, A. Ng, and C. Manning. “Parsing natural scenes and natural language with recursive neural networks." In Proc. ICML, 2011.

[24] S. Yaman, L. Deng, D. Yu, Y. Wang, and A. Acero. “An integrative and discriminative technique for spoken utterance classification.” In IEEE Transactions on Audio, Speech, and Language Processing, 2008.

[25] Y. Zhang, L. Deng, X. He, and A. Acero. “A novel decision function and the associated decision-feedback learning for speech translation.” In Proc. ICASSP, 2011.