Helioid at TREC-Legal 2011: Learning to Rank from Relevance Feedback for e-Discovery

Peter Lubell-Doughtie and Kenneth Hamilton

Helioid Inc., New York, NY, USA
{peter,kenneth}@helioid.com
http://www.helioid.com

Abstract. We present the results of applying a learning to rank algorithm to the 2011 TREC Legal dataset. The learning to rank algorithm we use was designed to maximize NDCG, MAP, and AUC scores. We therefore examine our results using the AUC and hypothetical F1 scores. We find query expansion and learning to rank improve scores beyond standard language model retrieval, however learning to rank does not outperform query expansion.

1 Introduction

Our approach begins with language modeling and then, as feedback from the user is received, we combine relevance feedback and learning to rank on the query level to improve result rankings using information from user interaction. This approach is based on the work in [5], which showed that by using relevance feedback in conjunction with learning to rank, result ordering could be improved beyond what is possible with only relevance feedback. We use learning to rank in order to automatically tune the amount and nature of relevance feedback specifically to the circumstance of a particular query. We apply learning to rank by first constructing a feature representation of our relevance feedback, and then using a pair-wise learning to rank algorithm which reorders documents based on their features and the features of judged documents.

In this paper we present experiments applying learning to rank with relevance feedback to the 2011 TREC Legal learning task. The framework our system used was designed to maximize AUC scores and we will therefore compare our system system using the AUC and hypothetical F1 scores, the same metric used in 2010 TREC Legal.

2 Related Work

The most commonly used query expansion method is the Rocchio algorithm [8]. It is based on the vector space representation of queries and documents and aims to select expansion terms from the relevant documents to create a query that can optimally distinguish between relevant and non-relevant documents. Typically, implementations of the Rocchio algorithm ignore non-relevant documents because excluding terms has been found to be problematic [16]. However, [16] find that negative feedback can be beneficial, especially for difficult queries. Our method avoids the problems of modeling negative feedback by using non-relevant documents in reordering, not query expansion.

Problems with sensitivity to parameter settings have led to substantial research in *adaptive relevance feedback*, where parameters such as the number of expansion terms, or the relative weight of expanded and original query terms are tuned using machine learning methods [7, 9]. Previous work considers the task of tuning these settings across queries (e.g., learning to predict the optimal weights for interpolation based on query and result set characteristics). Our work is complementary to these approaches, as it focuses on learning to rank for individual queries.

2.1 Additional Research

In other approaches, [3] apply latent semantic indexing to reduce dimensions, and [10] use kernel density estimation and logistic regression to model the training data. [12] use relevance feedback based upon the seed documents, and [2] use a Naive Bayes classifier to model the seed documents. The Naive Bayes classifier is known to have problems with unbalanced datasets [14, 15]. In contrast, we use a learning to rank algorithm which computes an optimized ranking using stochastic gradient descent. With this algorithm, called Ranking SVM, we are able to avoid problems of dataset balance.

The methods which have been successfully used to address the TREC Legal track are similar to those used in the TREC-CHEM track, which involves a prior art search task and is similarly recall oriented [6]. Given this overlap, we expect that applying our research to patent search will be a fruitful avenue for future work.

3 Method

The learning to rank from relevance feedback system learns from user feedback to build a model with which it reorders documents to better serve the user's information needs. In the interactive setting our approach models the retrieval task as a series of interactions between the retrieval system and the user. Figure 1 presents a high level overview of our search system. The user first submits a query, with which the system retrieves results. The system then returns these results to the user, who judges a subset of them. Next the system uses these judged documents to expand the query and reorder results through learning to rank. Finally, the reordered results are returned to the user, who is either satisfied and ends the search, or judges more results and continues the retrieval process.

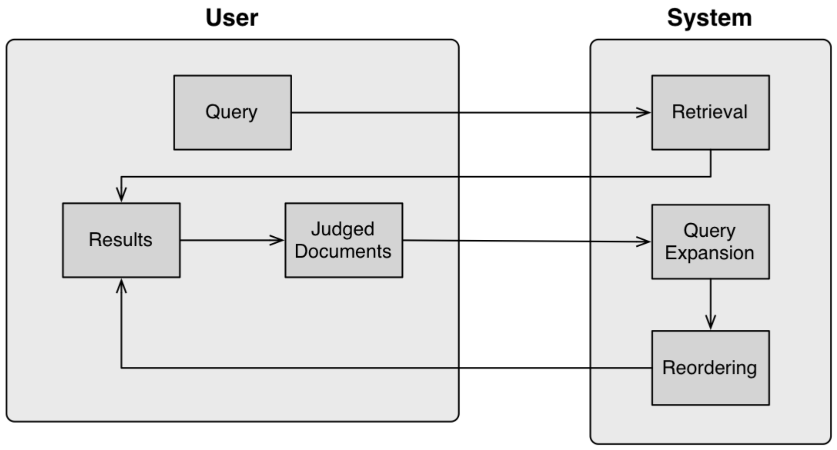

Fig. 1. A high level overview of our search system.

However, in the TREC Legal Learning task there is no interaction. In this case we use the complete set of judged documents as the relevance feedback set, and will compare this to performing retrieval when we have no relevance feedback. The main components of our search system are:

1. a retrieval function that ranks documents,
2. an expansion function that builds updated queries from judged documents,
3. a learning function that learns a document ranking using judged documents.

In the first step we index and retrieve documents with the Lemur (Indri 5.0)[1] search engine using language modeling. We run queries under the default retrieval model [11, 13], which uses Dirichlet smoothing with $\mu = 2500$.

3.1 Query expansion

We learn from judged documents on two levels by using query expansion (second step) in concert with learning to rank (third step). In query expansion we use the judged documents, $D_{\mathbf{q}}$, to build a modified query:

$$\tilde{\mathbf{q}} = \alpha \mathbf{q}_0 + (1 - \alpha) \sum_{\mathbf{d}_j \in D_{\mathbf{q}}} \mathbf{d}_j, \tag{1}$$

where $\mathbf{q}_0$ is the original query and α is an interpolation parameter. We retrieve documents with the expanded query $\tilde{\mathbf{q}}$, which provides us with a retrieval score per document. We will use these retrieval scores as a feature in learning to rank.

[1] `http://www.lemurproject.org`

3.2 Learning method

To learn a reordering of our retrieved documents we begin by projecting the judged documents into different feature spaces to create a feature vector $\mathbf{x}_i = \Phi(\mathbf{d}_i, \mathbf{q})$, where Φ extracts features from document $\mathbf{d}_i$ with respect to query $\mathbf{q}$. We use document judgements to build a set of preference pairs $\mathcal{P}$ such that $(i,j) \in \mathcal{P}$ iff document $\mathbf{d}_i$ is preferred to $\mathbf{d}_j$. We then minimize the objective function:

$$\frac{1}{2}||\mathbf{w}||^2 + C \sum_{(i,j)\in\mathcal{P}} \ell(\mathbf{w}^\top \mathbf{x}_i - \mathbf{w}^\top \mathbf{x}_j), \tag{2}$$

where C is a parameter that allows trading-off margin size against training error and ℓ is an appropriate loss function, in our case it is the squared hinge loss: $\ell(t) = \max(0, 1-t)^2$ [1, 4]. Minimizing Equation 2 gives us $\mathbf{w}$, the weighting vector over the feature space which minimizes the number of incorrectly ordered pairs of judged documents.[2]

We then project the top m documents from the corpus—as ranked by query $\mathbf{q}$—into the same feature space and use $\mathbf{w}$ to classify these documents:

$$\mathbf{y} = \mathbf{w}^\top \mathbf{X},$$

where $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_m]$ are feature vectors for documents $[\mathbf{d}_1, \ldots, \mathbf{d}_m]$, and each value $y_i \in \mathbf{y}$ is the predicted relevance of document $\mathbf{d}_i$. We sort the documents by their predicted relevance, as given by $\mathbf{y}$, and append the not-reordered documents ranked greater than m, to produce a reordering of the entire corpus.

We build features using the relevance feedback documents in multiple retrieval runs with different queries constructed by varying the number of expansion terms. The feature extractor Φ takes $P_\mathbf{q}$, the vector of judged documents from relevance feedback, r different expanded queries $\{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_j} | j \le r\}$, and the retrieval scores when using these expanded queries:

$$\mathbf{\Phi}\left(\begin{bmatrix} \{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_1}, P_\mathbf{q}, \langle s_1^{(1)}, \ldots, s_n^{(1)} \rangle\} \\ \vdots \\ \{\mathbf{q}_0 \circ t_1 \circ \cdots \circ t_{l_r}, P_\mathbf{q}, \langle s_1^{(r)}, \ldots, s_n^{(r)} \rangle\} \end{bmatrix}\right) = \begin{bmatrix} s_1^{(1)}, \ldots, s_m^{(1)} \\ \vdots \\ s_1^{(r)}, \ldots, s_m^{(r)} \end{bmatrix}, \tag{3}$$

where $\{l_j | j \le r\}$ indicates the number of terms used in expansion, e.g. if $l_j = 10$ we expand the query with 10 terms and $\langle s_1^{(j)}, \ldots, s_n^{(j)} \rangle$ are the retrieval scores when using this expanded query. These features are formed from the same judged document set, and thus the specific set of judged documents has a significant influence on the feature space and reordering, this implies that it is a more *exploitative* strategy.

However, this method has the practical disadvantage that it requires a retrieval run for each feature. For example, to test re-ranking with r features each iteration requires r retrieval runs: one to generate each additional feature for each expanded query. In practice this is not a problem because the retrieval runs can be parallelized.

[2] We use SGD-SVM as implemented in sofia-ml, `https://code.google.com/p/sofia-ml/`, which solves equation 2 with stochastic gradient descent.

4 Experiments

We use the judged documents in 2011 TREC Legal to evaluate three scenarios:

1. **lm** - standard language model retrieval using the original query.
2. **expansion** - query expansion with 20 terms
3. **learning** - our approach using learning to rank with $r = 4$ and $\mathbf{l} = \{5, 10, 15, 20\}$

In the *expansion* scenario we chose to use 20 expansion terms because experiments with the 2010 TREC Legal data found this setting to perform best. We further note that the system we used for evaluation is based on the 2010 TREC Legal dataset, in which the AUC score was the basis for measurement. Therefore we will present results of our learning to rank system based on both hypothetical F1 scores and AUC scorse.

5 Results and Discussion

Table 1 presents the hypothetical F1 scores for all queries and Table 2 shows the AUC scores. In the first columns we see that the baseline, language model retrieval, achieves an average score over all queries of 9.8 and 80.6 for hypothetical F1 and AUC scores respectively. Next, *expansion* outperforms *lm* with hypothetical F1 and AUC scores of 15.0 and 82.5 respectively. Our method, *learning*, also outperforms *lm* and is competitive with the relevance feedback run (*expansion*), with hypothetical F1 and AUC scores of 14.9 and 82.5, respectively.

Per query, the performance across all methods varies substantially. All our retrieval methods are based upon language modeling and we therefore expect that this variation is because some queries contain keywords that are more helpful for retrieval than others. From a cursory examination of the queries, we note that query 401, for which we achieve the highest scores across methods, concerns online services and contains the term `enrononline`, which is likely to be quite informative. However, query 403, on which performance is worst across methods, concerns the environmental impact of Enrons activities, a subject which the company would perhaps have an incentive to obscure and may therefore be less likely to be explicitly mentioned in email communications.

Table 1. Hypothetical F1 scores per query and averaged over all queries.

Query	**lm**	**expansion**	**learning**
401	18.3	**24.6**	**24.6**
402	7.8	**11.1**	11.0
403	3.2	9.0	**9.3**
AVG	9.8	**15.0**	14.9

Table 2. AUC scores per query and averaged over all queries.

Query	**lm**	**expansion**	**learning**
401	72.5	**76.0**	**76.0**
402	88.6	**89.0**	**89.0**
403	80.7	**82.6**	82.5
AVG	80.6	**82.5**	**82.5**

5.1 Comparison to Previous Research

Our results show that applying learning to rank does not perform substantially better than using query expansion alone. These results are in contrast to previous research using the TREC 2011 data, in which learning to rank substantially outperformed query expansion [5]. However, previous work used the more common information retrieval metrics of normalized discounted cumulative gain (NDCG) and mean average prevision (MAP), whereas the 2011 TREC Legal evaluation does not calculate these metrics.

There are also differences in the dataset. There are fewer, 3 versus 8, queries in the 2011 dataset compared to the 2010 dataset. Perhaps more significantly, there are a greater number of judged documents per query for the 2011 dataset in comparison with the 2010 dataset. On average there are about twice as many judged documents in 2011 and less variation in the number of judged documents per query. It is possible that the larger number of judged documents correlates with less stringent requirements on the degree of (non-)relevance necessary for a document to be marked as (non-)relevant. If this is the case, it would dilute the feature space and require adjustments in our learning to rank method.

6 Conclusions and Future Work

We have described our application of learning to rank from relevance feedback to the 2011 TREC Legal dataset. Interestingly, using this dataset learning to rank is not able to substantially outperform query expansion alone, although both learning to rank and query expansion outperform an approach based solely on language modeling. We speculate that learning to rank is not able to improve results because the judged documents are of a lower quality in the 2011 dataset as compared to the 2010 dataset, in virtue of the larger number of judged documents.

More work is in order to investigate why learning to rank does not outperform query expansion. We also note that the original research evaluated learning to rank with a number of different feature representations [5], and in this research we used only one feature representation. Future work applying additional feature representations to the the 2011 TREC Legal data can provide a clearer picture of the differences in performance between this dataset and the 2010 dataset.

Bibliography

[1] O. Chapelle and S. S. Keerthi. Efficient algorithms for ranking with svms. *Information Retrieval*, 13(3):201–215, 2010.

[2] A. Culotta, A. Liu, M. Cordover, B. Borden, and S. Strickland. IT-Discovery at TREC 2010 Legal. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[3] A. Garron and A. Kontostathis. Applying Latent Semantic Indexing on the TREC 2010 Legal Dataset. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[4] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[5] P. Lubell-Doughtie and K. Hofmann. Learning to rank from relevance feedback for e-discovery. In *ECIR*, 2012.

[6] M. Lupu, J. Huang, J. Zhu, and J. Tait. Trec-chem: large scale chemical information retrieval evaluation at trec. *SIGIR Forum*, 43:63–70, December 2009.

[7] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *CIKM '09*, pages 255–264, 2009.

[8] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[9] G. Pandey and J. Luxenburger. Exploiting session context for information retrieval - a comparative study. In *ECIR*, pages 652–657, 2008.

[10] D. P. Papadopoulus, V. S. Kalogeiton, and A. Arampatzis. DUTH does Probabilities of Relevance at the Legal Track. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[11] J. M. Ponte. A language modeling approach to information retrieval. Master's thesis, Amherst, MA, USA, 1998.

[12] S. Tomlinson. Learning Task Experiments in the TREC 2010 Legal Track. In E. M. Voorhees and L. P. Buckland, editors, *The Nineteenth Text REtrieval Conference Proceedings (TREC 2010)*. National Institute of Standards and Technology (NIST) the Defense Advanced Research Projects Agency (DARPA) and the Advanced Research and Development Activity (ARDA), Nov. 2010.

[13] H. R. Turtle. Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9:187–222, 1991.

[14] S. Verberne, H. van Halteren, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank QA data. In *Proceedings of the Learning to Rank workshop at SIGIR 2009*, pages 41–48, 2009.

[15] S. Verberne, H. Van Halteren, S. Raaijmakers, D. Theijssen, and L. Boves. Learning to Rank for Why-Question Answering. *Information Retrieval*, 2010.

[16] X. Wang, H. Fang, and C. Zhai. A study of methods for negative relevance feedback. In *SIGIR '08*, pages 219–226, 2008.