Feature Selection for Ranking using Boosted Trees

Feng Pan, Tim Converse, David Ahn, Franco Salvetti, Gianluca Donato
Microsoft / Powerset
475 Brannan St.
San Francisco, CA 94122
{fengpan, timcon, dahn, francosa, gdonato}@microsoft.com

ABSTRACT
Modern search engines have to be fast to satisfy users, so there are hard back-end latency requirements. The set of features useful for search ranking functions, though, continues to grow, making feature computation a latency bottleneck. As a result, not all available features can be used for ranking, and in fact, much of the time, only a small percentage of these features can be used. Thus, it is crucial to have a feature selection mechanism that can find a subset of features that both meets latency requirements and achieves high relevance. To this end, we explore different feature selection methods using boosted regression trees, including both greedy approaches (selecting the features with highest relative importance as computed by boosted trees; discounting importance by feature similarity and a randomized approach. We evaluate and compare these approaches using data from a commercial search engine. The experimental results show that the proposed randomized feature selection with feature-importance-based backward elimination outperforms greedy approaches and achieves a comparable relevance with 30 features to a full-feature model trained with 419 features and the same modeling parameters.

Categories and Subject Descriptors
H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval – *Selection process*.

General Terms
Algorithms, Experimentation, Performance

Keywords
Information retrieval, feature selection, boosted trees

1. INTRODUCTION
Ranking search results is essential for information retrieval and Web search. Search engines need to not only return highly relevant results, but also be fast to satisfy users, so there are always hard back-end latency requirements. With the set of features useful for ranking functions continuing to grow, feature computation has become a latency bottleneck. As a result, not all available features can be used for ranking, and in fact, much of the time, only a small percentage of these features can be used. Thus, it is crucial to have a feature selection mechanism that can find a subset of features that both meets latency requirements and achieves high relevance.

Despite its importance, not much work has been done on feature selection for ranking search results. Among the most closely related work is [6], which proposes proposed an efficient greedy feature selection method for ranking that finds the features with maximum total importance scores and minimum total similarity scores. However, as we discuss in Section 4.2, this approach cannot select query features (e.g., query length). Only a small number of features were extracted in their experiments (i.e., 44 and 26 features respectively for the two experiments), which makes it hard to evaluate whether a feature selection method can reduce the feature dimension significantly with preserved accuracy/relevance.

Boosted trees (and boosting algorithms in general) have been used widely as a learning algorithm for ranking search results, for example, McRank [9], LambdaMART [4], RankBoost [12], and AdaRank [10]. However, to our knowledge, there is little work on using boosted trees for feature selection in ranking. Although boosted stumps (i.e., single-layer boosted trees) by RankBoost have been used for feature selection in ranking [13], this approach has similar drawbacks as [6]: query features cannot be selected and contributions from feature interaction are completely ignored. In this paper, we explore different feature selection methods using boosted trees, including both greedy and randomized approaches. Specifically, in Section 2 we describe why we use boosted trees for both ranking and feature selection and how relative importance of features is computed. The data and representation for experiments are described in Section 3. The experimental results of greedy feature selection approaches are shown in Section 4. A randomized feature selection algorithm is proposed in Section 5, which performs a randomized feature-importance-based backward elimination, and the experimental results show that it outperforms the greedy approaches.

2. BOOSTED TREES AND RELATIVE IMPORTANCE OF FEATURES
The boosted trees algorithm we used for ranking and feature selection is the gradient boosting machine [1]. There are many advantages of using boosted trees as a learning algorithm for ranking. For example, no normalization is needed when using

CIKM'09, November 2–6, 2009, Hong Kong, China.

different types of data (e.g., categorical and count data); trading off runtime efficiency and accuracy (i.e., relevance for search) can be easily achieved by truncating the number of trees used in the boosted trees model.

From the perspective of feature selection, a more interesting property of the boosted trees is that (a greedy) feature selection already happens in the algorithm when selecting splitting features (e.g., for regression trees, splitting features and splitting points are found to minimize the squared-error loss for any given partition of the data). Moreover, as a byproduct, a sorted list of relative importance of features (i.e., a feature importance list) is automatically generated for each boosted trees model.

Friedman [1] generalizes the relative influence of a feature x_j for a single decision tree to boosted trees as an average over all the trees

$$\hat{J}_j^2 = \frac{1}{M}\sum_{m=1}^{M}\hat{J}_j^2(T_m)$$

where M is the number of trees. For each tree the relative importance is calculated as

$$\hat{J}_j^2(T) = \sum_{t=1}^{L-1}\hat{I}_t^2 1(v_t = j)$$

where the summation is over the internal nodes t of a L-terminal node tree T, v_t is the splitting feature associated with node t, and $\hat{I}_t^2$ is the corresponding empirical improvement in squared-error as a result of the split.

3. Data and Representation

The experimental results we report in this paper used the data from a commercial search engine for Wikipedia articles. There are 479 features in total, including query features (e.g., query length), document features (e.g., document length), and match features (e.g., BM25 [3]). As a preprocessing step, we removed 60 features with the worst cost-effectiveness (i.e. very expensive to compute and ranked low in the feature importance list). Thus 419 features are actually used in the feature selection experiments. This is one use of the feature importance list to reduce the search space of features.

The data comprises 3800 queries and 79811 results. It is divided into a training set (3000 queries) and a testing set (800 queries). 5-fold cross-validation are used for deciding the modeling parameters for the boosted trees ranker. All the models in the following experiments were trained using the same modeling parameters.

Each result (a query-URL pair) was judged on a 5 point scale from 0 (not relevant) to 3 (very relevant) at least 5 times by annotators from Amazon's Mechanical Turk[1], an online labor market where workers are paid small amounts of money to complete human intelligence tasks. Judgments from automatically detected scammers are removed. [8] shows a high agreement between Mechanical Turk non-expert annotations and existing gold-standard labels provided by expert labelers for five natural language processing tasks. Multiple labeling has also been demonstrated useful for improving data quality of annotations from non-experts [5].

The target value of a result is an averaged judgment from multiple annotators. Thus, the ranking problem of search results is cast as a regression problem in machine learning. Squared-error is used as the loss function for regression in the boosted trees ranker.

4. GREEDY FEATURE SELECTION

In this section we describe two greedy feature selection approaches and their experimental results. Both of them do forward selection, a wrapper methodology for feature selection that utilizes the learning algorithm as a black box to score subsets of features according to their predictive power [11].

4.1 Selecting Features with Highest Relative Importance

The most simple and straightforward feature selection when using boosted trees as the learning algorithm is to directly select the top features with the highest relative importance scores as calculated in Section 2. Figure 1 shows a curve of NDCG@5 scores of models trained with different number of top features from the feature importance list (up to top 40 features). The feature importance list was generated by a boosted trees model trained with all the 419 features.

This feature selection is a kind of forward selection, which is usually computationally more efficient than the opposite wrapper approach, i.e., backward elimination, especially when the final selected set of features is significantly less than the full set of features. In fact, this is true for our experiments – due to the latency requirement, depending on the individual feature computation and extraction time, we normally can only afford up to 40 features in our production ranking function out of 419 features.

However, it is argued that weaker subsets are found by forward feature selection because the relative importance of features is assessed in the context of features that may not be included [11]. This argument is plausible in the context of boosted trees since feature interactions are explicitly taken into consideration when computing the relative importance of features, as defined in Section 2. Concretely, a feature that is ranked highly in the feature importance list for a model trained on the full set of features may turn out to be relatively unimportant in a smaller model that omits the other features with which it interacts. A proposed randomized feature selection algorithm described in Section 5.2 tries to perform backward elimination in a more efficient and randomized fashion.

Despite its simplicity, directly selecting the top features from the feature importance list sets a pretty high baseline standard. For example, the model trained with the top 30 features achieves a NDCG@5 score of 86.74, which is already very close to the model trained with all the 419 features (87.01). The peak of the top 40 features occurs at 28 features with a relevance score of 86.87.

[1] https://www.mturk.com

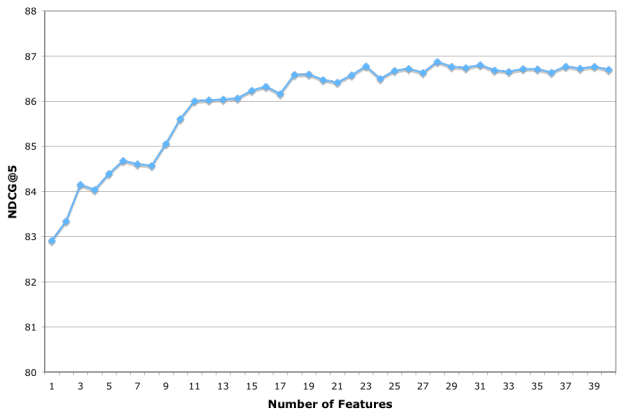

Figure 1. Feature selection with highest relative importance

4.2 Discounting Importance by Feature Similarity

Inspired by the work in [6], we try to discount the feature importance scores by feature similarity, i.e., selecting important features that are less similar to each other. In [6], it tries to select a subset of features with largest total importance scores and smallest similarity scores. However, we are very different in how to compute those scores and also how to combine them.

Feature importance scores are computed in a model-independent way in [6], i.e., each feature is used individually to rank the results, and a NDCG or a loss function score is calculated as their feature importance scores. This approach, however, can't assign an importance score to any query features (e.g., query length) since all the results rank the same for a given query.

We, instead, take a model-dependent approach, i.e., directly using the feature importance scores automatically generated by the boosted trees ranker. In this way, query features are assigned with importance scores appropriately accordingly to their contributions to the overall error reduction by interacting with other features in the model. Similarly, [6] also uses the ranked results by each feature to compute feature similarity, i.e., the correlation score between the ranked lists is used as their similarity score. Again, query features can't be handled in this approach. To address this issue, we instead use the correlation score between the vectors of feature values as their similarity score.

While features are selected from the top of the feature importance list, the feature importance list is updated iteratively, i.e., after a feature is selected from the top of the list, the importance scores of the rest of the features are updated by dividing their similarity score with respect to the newly selected feature. Since the similarity score is in the range from 0 to 1, the feature importance scores are effectively boosted for those features that are dissimilar to the top features, and the more dissimilar, the more importance score boost. To prevent boosting too much the importance score of a feature that is very dissimilar to top features, a parameter of max boost is set, which controls how aggressively a low similarity feature can boost its importance score.

Figure 2 shows a comparison between different max boost values ranging from 2 (very conservative feature similarity discounting) to 50 (very aggressive feature similarity discounting) and original feature importance list without feature similarity discounting The experiments indicate that higher max boost values (e.g., 10 and 50) perform significantly worse than lower max boost values (e.g., 2), and the original feature importance list without similarity discounting perform slightly better than max boost 2 for most of cases except with less than 10 features and more than 32 features. Basically, these two curves look very close, which is not surprising – with a low similarity discounting factor (i.e., max boost), the feature importance list is not changed too much from the original one. This result seems to indicate that boosted trees are already pretty good at selecting between similar or correlated features.

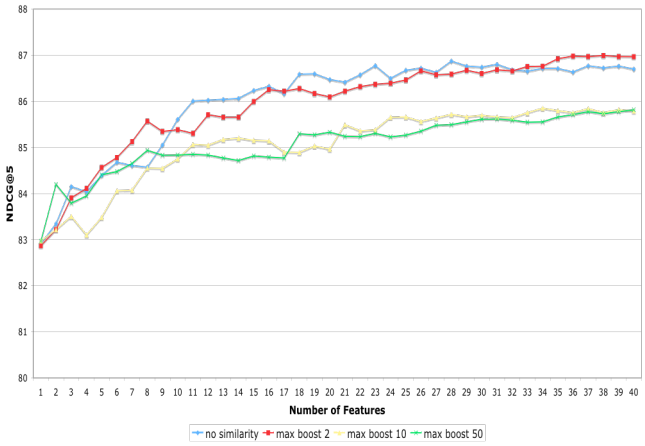

Figure 2. Feature selection with vs. without similarity discounting

5. RANDOMIZED FEATURE SELECTION

The experimental results of the greedy feature selection methods demonstrate that it's hard for greedy approaches to outperform the baseline that directly takes the top features from the feature importance list generated by the boosted trees ranker. Since we cannot afford to search the entire feature space exhaustively, randomized approaches become good options to search the space more extensively than the greedy approaches.

5.1 Randomized Feature Selection with Feature-Importance-Based Backward Elimination

We propose a randomized feature selection method. It does backward elimination in an efficient and randomized fashion (as described in Section 4.1, it is argued that backward elimination usually generates a better subset of features than forward selection, but with higher computational cost), and feature importance scores are incorporated in deciding in each round which features to eliminate.

Here is a brief outline of the algorithm:

1. Randomly partition all the features into disjoint subsets, each of which contains 30 features[2] (any leftover features will move directly to the next round).

[2] For comparison purpose, we assume the target number of features to select is 30.

2. Train a boosted trees model with each subset of features.
3. A percentage (called a "feature elimination rate") of features are removed from each round, which controls how long the search goes.
4. The number of features to remove from each subset is determined by their NDCG@5 scores: the higher the score is, the fewer features will be removed from that subset. Features are removed from the bottom of the feature importance list of each subset.
5. Merge all the remaining features together. Repeat step 1 to 4 until the number of features left is no more than 30.

There are some extra details about how the features are eliminated. First, at least one feature is removed from each subset to ensure the number of total features reduces from one round to the next. This can be changed to be more conservative, for example, only requiring at least one feature is removed from each round, i.e., likely to be the bottom one from the subset with the worst relevance score.

Second, since many subsets of features receive pretty close NDCG@5 scores, in order to be able to distinguish them more and have different number of features removed, instead of using the raw NDCG@5 scores, the differences from the best relevance score for that round is used. For example, if the NDCG@5 scores for a round are {70, 75, 78}, then the updated scores used for deciding how many features to remove from each subset become {8, 3, 0}, and the bottom features will be removed in proportion to those updated scores after add-one smoothing.

The experiment with 1.25% feature elimination rate achieved the best NDCG@5 score of 86.88 (Figure 3), which is already higher than the baseline using top 30 features from the importance list (86.74), and comparable to the peak with 28 features (86.87).

In fact, even with a higher feature elimination rate (e.g, 50%, 25%), the best relevance achieved (86.82 with 50%, and 86.81 with 25%) already outperforms the baseline using top 30 features from the importance list. One big advantage is that the total number of samples is significantly fewer with high elimination rates, and thus the selection takes much less time to finish.

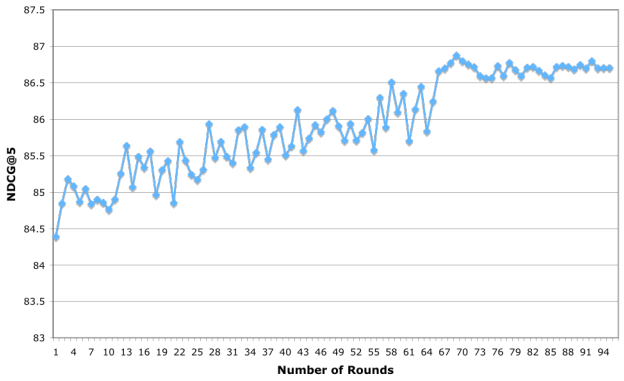

Figure 3. Randomized feature selection using all 419 features with 1.25% feature elimination rate

6. CONCLUSIONS

In this paper, we have presented our work on feature selection for ranking search results using boosted regression trees. We compared and evaluated three different feature selection methods, including both greedy and randomized approaches. We proposed a randomized method with feature-importance-based backward elimination. The experimental results show that it outperformed the greedy approaches, and with 30 features, it achieved a comparable relevance to a full-feature model trained with 419 features and the same modeling parameters.

7. REFERENCES

[1] J. H. Friedman. 2001. Greedy Function Approximation: A Gradient Boosting Machine. Annals of Statistics 29(5):1189-1232.

[2] Y-K Wang, K-C Fan, J-T Horng. 1997. Genetic-based search for error-correcting graph isomorphism. IEEE Transaction on Systems, Man and Cybernetics. 27(4): 588-597.

[3] S. Robertson. Overview of the okapi projects. 1997. Journal of Documentation, 53(1): 3-7.

[4] Q. Wu, C. J. C. Burges, K. M. Svore and J. Gao. 2008. Ranking, Boosting, and Model Adaptation. . Microsoft Research Technical Report MSR-TR-2008-109.

[5] V. S. Sheng, F. Provost and P.G. Ipeirotis. 2008. Get Another Label? Improving Data Quality and Data Mining Using Multiple, Noisy Labelers. In Proceedings of the Knowledge Discovery and Data Mining Conference (KDD).

[6] X. Geng, T-Y Liu, T. Qin and H. Li. 2007. Feature Selection for Ranking". In Proceedings of the International Conference on Research and Development in Information Retrieval.

[7] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen and G. Sun. 2007. A General Boosting Method and its Application to Learning Ranking Functions for Web Search. In Proceedings of the Annual Conference on Neural Information Processing Systems (NIPS).

[8] R. Snow, B. O'Connor, D. Jurafsky and A. Y. Ng. 2008. Cheap and Fast – But is it Good? Evaluating Non-Expert Annotations for Natural Language Tasks. In Proceedings of the Conference on Empirical Methods on Natural Language Processing (EMNLP).

[9] P. Li, C. J. C. Burges and Q. Wu. 2007. Learning to Rank Using Classification and Gradient Boosting. Microsoft Research Technical Report MSR-TR-2007-74.

[10] J. Xu and H. Li. 2007. AdaRank: A Boosting Algorithm for Information Retrieval. In Proceedings of the International Conference on Research and Development in Information Retrieval (SIGIR).

[11] I. Guyon and A. Elisseeff. 2003. An Introduction to Variable and Feature Selection. Journal of Machine Learning Research 3 (2003) 1157-1182.

[12] Y. Freund, R. Iyer, R. E. Schapire and Y. Singer. 2003. An Efficient Boosting Algorithm for Combining Preferences. Journal of Machine Learning Research 4 (2003) 933-969.

[13] N-Y Xu, X-F Cai, R. Gao, L. Zhang and F-H Hsu. 2007. FPGA-based Accelerator Design for RankBoost in Web Search Engines. In Proceedings of the International Conference on Field-Programmable Technology (ICFPT).