Expected Divergence based Feature Selection for Learning to Rank

Parth Gupta   Paolo Rosso
Natural Language Engineering Lab - ELiRF
Department of Information Systems and Computation
Universitat Polit cnica de Val ncia, Spain
`http://www.dsic.upv.es/grupos/nle`
`{pgupta,prosso}@dsic.upv.es`

Abstract

Feature selection methods are essential for learning to rank (LTR) approaches as the number of features are directly proportional to computational cost and sometimes, might lead to the over-fitting of the ranking model. We propose an expected divergence based approach to select a subset of highly discriminating features over relevance categories. The proposed method is evaluated in terms of performance of standard LTR algorithms when trained with reduced features over a set of standard LTR datasets. The proposed method leads to not significantly worse, and in some cases, significantly better performance compared to the baselines with as few features as less than 10%. The proposed method is scalable and can easily be parallelised.

Title and Abstract in another language, L_2 (optional, and on same page)

श्रेणी अधिगम के लिए अपेक्शित विचलन आधारित नक्श चयन

श्रेणी अधिगम (learning to rank) प्रक्रिया के लिए नक्श चयन पद्धतियाँ महत्वपूर्ण हैं, क्योंकि संगणात्मक मूल्य, नक्श संख्या से समानुपाती है और कभी कभी नक्श संख्या रेंकिंग मॉडल की ओवर-फ़िटिंग को प्रेरित करता है। हम नक्श चयन के लिए अपेक्शित विचलन आधारित पद्धति का प्रस्ताव करते हैं जो प्रासंगिक वर्गों में अति विवेकी नक्श उपगण का चयन करती है। प्रस्तावित पद्धति का मान्य श्रेणी अधिगम संग्रह पर अल्प नक्शों से प्रशिक्शित मान्य श्रेणी अधिगम कलन गणितों (algorithms) के संपादन से मूल्यांकन किया गया है। प्रस्तावित पद्धति आधार रेखाओं की तुलना में सिर्फ 10% नक्शों के उपयोग से not significantly worse और कुछ किस्सों में significantly बेहतर प्रदर्शन दिखा रही है। प्रस्तावित पद्धति स्केलेबल है और आसानी से समान्तर चलाई जा सकती है।

Keywords: feature selection, information retrieval, ranking.

Keywords in L_2: नक्श चयन, सूचना पुनः प्राप्ति, रेंकिंग.

---

1 Introduction

Ranking is one of the most important modules of Information Retrieval (IR) systems. Unsupervised ranking models like BM25 okapi and language models have power to rank documents with limited number of features such as term frequency (TF), inverse document frequency (IDF), document length (DL). Although they can rank with speed and without the need of labeled relevance information, they are quite restrictive for the incorporation of more features such as age, link information in web graph, click information etcetera of the document. Elimination of such features from ranking might prove to be a big limitation in the rapidly increasing and annotation rich Web. To overcome this limitation, the research community has posed the ranking problem in machine learning framework and referred as learning to rank (LTR). LTR is a supervised setting of ranking in the IR system where, each document for the given query is represented as a feature vector to which, the ranking function assigns a score. The ranking function is trained on a prelabelled training data.

Over the time, the number of features used in the learning to rank has increased drastically. Although increasing number of features induces more information for the ranking algorithms, it is directly related to the computational complexity and to some extent the over-fitting of the ranking model in some cases. As a result, the attempts to reduce the dimensionality of the feature vector subsequently started (Geng et al., 2007; Pan et al., 2009; Dang and Croft, 2010).

Among a few approaches of feature selection for LTR, Geng et al. (2007) proposed an efficient greedy feature selection method for ranking that finds the features with maximum total importance scores and minimum total similarity scores. The greedy search algorithm over an undirected graph of features was employed to solve the optimization problem. In contrast, Dang and Croft (2010) used best first search to come up with subsets of features and coordinate ascent to learn the weights for those features. This approach, feature selection - best first search (`FS-BFS`), has recently shown to outperform the greedy approach, hence we use it as one of the baselines to compare with. Pan et al. (2009) used boosting trees with randomized and greedy approach, where th *wrapper* approach was taken with forward selection and backward elimination.

In our approach, the subset of features are selected based on their expected divergence over the relevance classes and the importance of features are estimated by the evaluation scores produced individually by the features. We use Kullback-Leibler (KL) divergence to estimate the divergence and adapt it to make it more suitable for the ranking. The results with the proposed method are reported on a set of standard LTR datasets with three state-of-the-art LTR algorithms RankSVM (Herbrich et al., 2000), RankBoost (Freund et al., 2003) and LamdaMart (Wu et al., 2010). We use the performance of LTR algorithms when learnt with all features (`WAF`) as another baseline. The performance achieved with the proposed feature selection method is statistically similar to the baselines and in some cases the performance is significantly improved with very few features as 10%. Moreover, the proposed algorithm can easily be parallelised.

We describe the details of the proposed method in Section 2. In Section 3 we describe the experimental setup and the results with analyses are presented subsequently in Section 4. Finally, we end the discussion with concluding remarks in Section 5.

2 Method

The feature selection methods of type *filter*, as defined in Guyon and Elisseeff (2003), computes the score of each feature as a preprocessing step and the subset of features are selected based on the scores assigned. In contrast to *filter* methods, *wrapper* methods use the learning algorithms to assign scores to the features. We opt for a *filter* approach and refer it as feature selection - expected divergence (`FS-ED`), while `FS-BFS` is a *wrapper* approach.

The proposed method has two components: (*i*) the importance of the features defined as $s(f_i)$ and, (*ii*) the expected divergence of the features defined as $d(f_i)$. The goal of the method is to score each feature $f_i \in F$, where F is the set of all features and $|F| = n$. We pose the feature selection method as a maximization problem of selecting top k features from F where, the score of a feature $\psi(\cdot)$ is calculated as shown in Eq. 1.

$$\psi(f_i) = s(f_i) + d(f_i) \tag{1}$$

As reported in Geng et al. (2007), the feature importance $s(f_i)$ derived from evaluation scores and learning algorithms lead to statistically similar results. Hence, we opt for the evaluation measure, NDCG@10, to estimate the importance of an individual feature. The evaluation score for the queries in the training data using a particular feature value individually to rank documents is considered as the importance score $s(f_i)$.

In order to estimate the divergence of a feature over the relevance classes, we use KL divergence. KL divergence has successfully been used for the feature selection methods for classification problems (Coetzee, 2005; Schneider, 2004). Because of the intuitive differences between the classification and ranking, we adapt and call it as expected divergence which, to the best of our knowledge, is novel. Classes in ranking are ordinal relevance levels, while they are unordered categories in case of classification. Hence, we boost the divergence of a feature over distant relevance classes and calculate the expected divergence. For example, consider a 5-scale relevance system with relevance classes $r_i \in R$ where $i \in \{0, 1, .., 4\}$, the divergence of a feature over r_0 and r_4 is far more important than that over r_0 and r_1. The expected divergence of a feature is calculated as shown in Eq. 2.

$$d(f_i) = \sum_{m=0}^{|R|-1} \sum_{n=m+1}^{|R|-1} (n - m) * \mathrm{div}(f_i^{r_m}, f_i^{r_n}) \tag{2}$$

where,

$$\mathrm{div}(f_i^{r_m}, f_i^{r_n}) = \frac{1}{2} d_{KL}\left(\hat{f}_i^{r_m} || \hat{f}_i^{avg}\right) + \frac{1}{2} d_{KL}\left(\hat{f}_i^{r_n} || \hat{f}_i^{avg}\right) \tag{3}$$

Eq. 3 is the Jensen-Shannon divergence where, $\hat{f}_i^{r_m}$ is the estimated probability density function (PDF) using kernel density estimation (KDE) of i^{th} feature over relevance class r_m learnt from the training data and estimated on the validation data as shown in Eq. 4, $\hat{f}_i^{avg}$ is an average over both relevance classes and d_{KL} is KL divergence.

$$\hat{f}_i = \frac{1}{Mh} \sum_{j=1}^{M} \phi\left(\frac{x - x_j}{h}\right) \tag{4}$$

where, M is the total number of samples in the training data, x_j and x refers to the value of i^{th} feature in the training and validation data respectively , h is the bandwidth which is estimated using *Silverman's rule of thumb* (Bernard, 1986) and the kernel is chosen as standard normal distribution (ϕ). The complete feature selection method is described in Fig. 1.

```
$T$ = training data
$V$ = validation data
$\vec{\psi}$ = weight vector of features
$F, F_k$ = feature sets, all and top-$k$ respectively
for each $f_i \in F$
    $\psi(f_i) = 0$     /* Initialise the weights */
end for
for each $f_i$
    $s(f_i)$ = evaluation score over $T$
    for each $r_i \in R$
        estimate $PDF(f_i)$ over $r_i$ from $T$ using KDE
    end for
    for $i = 0$ $to$ $|R| - 1$
        for $j = i + 1$ $to$ $|R| - 1$
            estimate JS div. of $f_i$ over $r_i$ and $r_j$ from $V$
        end for
    end for
    calculate $d(f_i)$ as show in Eq. 2
    $\psi(f_i) = s(f_i) + d(f_i)$
end for
sort $\vec{\psi}$
for $i = 1$ $to$ $k$
    add $f_i$ in $F_k$
end for
RETURN $F_k$
```

Figure 1: Feature selection procedure.

3 Experimental Setup

In order to compare the proposed method with the baselines, we use the performance evaluation of three state-of-the-art LTR algorithms when trained with selected features on four standard LTR datasets. We use NDCG@10 as metric and five-fold cross valiadation. NDGC@10 estimates the quality of ranking especially in the graded multi-scale relevance level setting (J rvelin and Kek l inen, 2002). Each ranking method is trained over training set and the model that performs best on validation set is used for testing in each fold.

3.1 Ranking Methods

3.1.1 RankSVM

RankSVM is a widely used pairwise LTR algorithm proposed in Herbrich et al. (2000). At first, the training data is transformed to make the pairs of correctly and incorrectly ranked documents, then an SVM model is trained to learn the weight vector $\vec{w}$. We used the publicly available implementation of RankSVM [1] to train the model on training data and choose the

[1] `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

parameters which maximizes the performance on the validation data. We use linear kernel, $epsilon = 0.001$ and loop over $[0.00001, 10]$ with step of x2 to estimate C.

3.1.2 RankBoost

RankBoost (Freund et al., 2003) is another popular pairwise LTR algorithm based on boosting technique. The boosting algorithm uses weak rankings to update the weights of the pairs. The weights of the correctly ranked instances are decreased and that of incorrectly ranked are increased to give them more importance in the next round. Finally, a linear combination of weak rankers is produced. We used a publicly available implementation of RankBoost[2] and train the model until no performance change is observed for 100 iterations.

3.1.3 LambdaMART

LambdaMART (Wu et al., 2010) uses gradient boosting, to optimize a ranking cost function, which produces an ensemble of regression trees. The final model can be seen as a weighted combination of such trees as shown in Eq. 5 where, N is the total number of regression trees and α_i is the weight associated with i^{th} tree.

$$F_N(x) = \sum_{i}^{N} \alpha_i * f_i(x) \tag{5}$$

More details, about LambdaMART can be found in Burges (2010). We used a publicly available implementation of LambdaMART[2] with following mentioned parameters, $\#\ of\ trees = 1000$, $learning\ rate = 0.1$ and $\#\ of\ tree\ leaves = 10$.

3.2 Datasets

We conduct the experiments on three standard LTR datasets: (i) OHSUMED, (ii) Letor 4.0 and (iii) Yahoo!. OHSUMED (Hersh et al., 1994) is a part of Letor 3.0 and contains documents from the MEDLINE, a corpus of medical publications. This corpus contains 106 queries, 3 levels and 45 features. The Letor 4.0 dataset is created from the Gov-2 document collection which contains roughly 25 million Web pages. It contains two query-sets MQ2007 and MQ2008 corresponding to years 2007 and 2008 editions of TREC Million Query track[3]. The Letor 4.0 has in total 2476 queries, 3 relevance levels and 46 features. We used Yahoo! SET 2 query-set which contains the LTR data of the commercial search engine and has 6330 queries, 5 relevance levels and 699 features.

Although the dimensionality of features in OHSUMED and Letor 4.0 is less than 50, we consider necessary to report results on these datasets. Based on the feature analysis presented in Geng et al. (2007), features importance in OHSUMED and .Gov datasets are highly different. Moreover, Information Gain and CHI based $filter$ approaches perform quite differently on them, hence we opt to investigate the stability of the proposed method on these datasets too.

[2] http://people.cs.umass.edu/~vdang/ranklib.html
[3] http://ciir.cs.umass.edu/research/million/

3.3 FS-BFS

The FS-BFS is a *wrapper* based approach of feature selection for ranking (Dang and Croft, 2010). The method partitions the F into non-overlapping k subsets and learns a ranking model which maximizes the performance over that subset of features. Best first search is used on the undirected graph of features to extract subsets and the weights of the features are learnt using coordinate ascent. Each best subset will be a weighted combination of the original features in the subset and will represent a completely new feature. We use the parameters as defined in (Dang and Croft, 2010).

4 Results and Discussion

Fig. 2 represents the performance evaluation of RankSVM, RankBoost and LamdaMART when trained with FS-ED, FS-BFS and WAF. We did the significance TTest of the results where we consider the p-value < 0.05 for statistical significance. As can be noticed from the figures, FS-ED performs statistically similar to the baselines in all the cases and in some cases it significantly outperforms the baselines. FS-BFS is a very computation intensive model and it was impractical to optimize the best first search over 699 features graph of a large dataset on our server. Therefore, the the results on Yahoo! dataset are not available with FS-BFS[4]. Here, we would like to mention that, in FS-ED, the scoring of a feature does not depend on the other features' scores so it can easily be parallelised for individual features as can be noticed from Fig. 1. On the contrary, FS-BFS can not be parallelised as the optimization of the consecutive subset depends on the prior best subset. Just to mention, running FS-ED on our server with parallelisation on 8 processors produced results for Yahoo! dataset in ∼1.5 hours under normal CPU load. We would also like to mention, the selection method of features presented in Geng et al. (2007) also depend on other features similarity with it, hence making it much computation intensive when running for large datasets like Yahoo!. Table 1 reflects the number of features used to produce the best results. It is noticeable that in some cases FS-ED was able to produce the best results using less than 10% of total features on all datasets.

FS Method	O	M7	M8	Y	R. Method
FS-ED	15	**3**	**3**	**50**	RankSVM
	4	15	15	75	RankBoost
	20	10	20	75	LambdaMART
FS-BFS	**6**	9	12	-	RankSVM
	12	**7**	**7**	-	RankBoost
	14	10	**7**	-	LambdaMART
WAF	45	46	46	699	ALL

Table 1: The number of features used to obtain the results reported in Fig. 2 with different feature selection strategies. O, M7, M8 and Y refer to OHSUMED, MQ2007, MQ2008 and Yahoo! respectively.

The proposed method exhibit similar behaviour on datasets like OHSUMED, MQ2007 and MQ2008 while, an interesting behaviour is noticed on Yahoo! dataset. FS-ED achieved more than 5 point gain in NDCG@10 for RankSVM with only 50 features while performed relatively worse with RankBoost and LambdaMART. To understand this phenomenon better, we analysed the distribution of the top and last features obtained using FS-ED over the relevance classes.

[4] We used Intel Xeon CPU E5520 @ 2.27GHz with 4 cores, 8 processors and 12GiB memory. We ran FS-BFS for around 7 days but did not notice any progress.

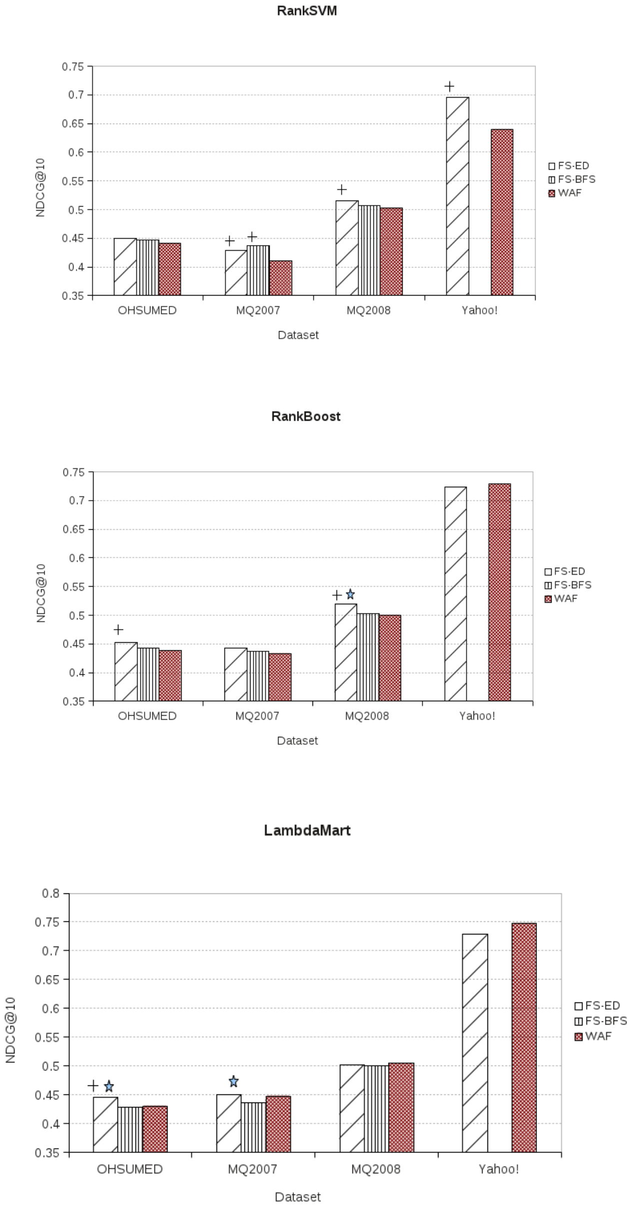

Figure 2: Performance evaluation of feature selection schemes on different datasets with RankSVM, RankBoost and LambdaMART. $+$ and $\star$ indicate statistical significance with WAF and the other FS strategy respectively.

The analysis is presented in Fig. 3. It is noticeable that, the top features better discriminate between the relevance classes and exhibit high divergence over distant relevance levels. Moreover, the expected divergence component minimizes the weight of those features which are least discriminative and in turn, might prove to be ambiguous for some of the ranking models.

We consider this as the main reason where FS-ED performs better compared to WAF. Because of this characteristic, the large margin classifier based ranking models like RankSVM are benefited more compared to the weak learner based models like RankBoost and LambdaMART. The weak learner based models can easily minimise the importance of the less discriminative features and hence the performance does not rapidly deteriorate as compared to large margin classifiers based models. This phenomenon makes the proposed method much efficient and important for the large-margin classifier based rankers which can clearly be noticed from statistical significance of FS-ED over WAF across the datasets for RankSVM. The method is quite robust, as the feature importance component captures the linearity of features over relevance classes while the expected divergence component enables the method to capture the non-linearity.

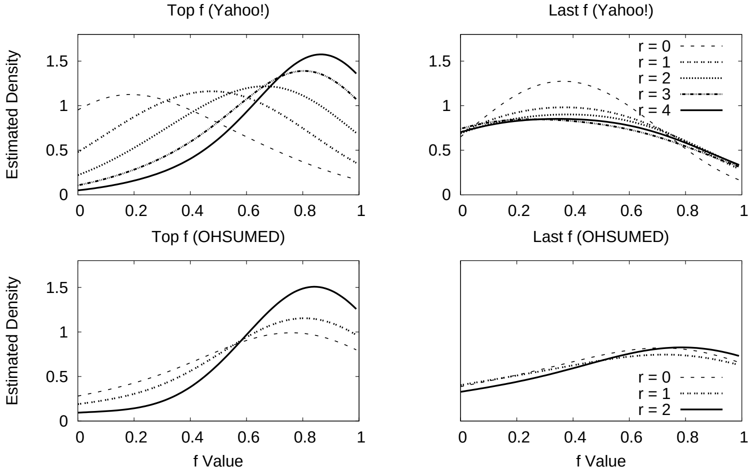

Figure 3: The estimated density of the top and last features according to FS-ED over relevance classes (r) on Yahoo! and OHSUMED. X-axis denote values a feature can take. The features having constant or zero value for all the queries are excluded.

5 Remarks

We proposed an expected divergence based feature selection method for learning to rank. The method is very efficient and can be parallelised. The proposed method leads to not significantly worse, and in some cases, significantly better performance compared to the baselines with as few features as less than 10% on a set of standard datasets and state-of-the-art LTR algorithms. We analysed the selected features over the relevance classes and exhibit that large margin classifier based ranking models can greatly benefit from the selection method.

References

Bernard, S. (1986). *Density estimation for statistics and data analysis (Chapman & Hall/CRC Monographs on Statistics & Applied Probability)*. Chapman and Hall/CRC, 1 edition.

Burges, C. J. (2010). From ranknet to lambdarank to lambdamart: An overview. In *Microsoft Research Technical Report MSR-TR-2010-82*.

Coetzee, F. M. (2005). Correcting the kullback-leibler distance for feature selection. *Pattern Recogn. Lett.*, 26(11):1675–1683.

Dang, V. and Croft, B. W. (2010). Feature selection for document ranking using best first search and coordinate ascent. In *SIGIR Workshop on Feature Generation and Selection for Information Retrieval,*, SIGIR '10.

Freund, Y., Iyer, R., Schapire, R. E., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *J. Mach. Learn. Res.*, 4:933–969.

Geng, X., Liu, T.-Y., Qin, T., and Li, H. (2007). Feature selection for ranking. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 407–414, New York, NY, USA. ACM.

Guyon, I. and Elisseeff, A. (2003). An introduction to variable and feature selection. *J. Mach. Learn. Res.*, 3:1157–1182.

Herbrich, R., Graepel, T., and Obermayer, K. (2000). *Large margin rank boundaries for ordinal regression*. MIT Press, Cambridge, MA.

Hersh, W., Buckley, C., Leone, T. J., and Hickam, D. (1994). Ohsumed: an interactive retrieval evaluation and new large test collection for research. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '94, pages 192–201, New York, NY, USA. Springer-Verlag New York, Inc.

J rvelin, K. and Kek l inen, J. (2002). Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446.

Pan, F., Converse, T., Ahn, D., Salvetti, F., and Donato, G. (2009). Feature selection for ranking using boosted trees. In *Proceedings of the 18th ACM conference on Information and knowledge management*, CIKM '09, pages 2025–2028, New York, NY, USA. ACM.

Schneider, K.-M. (2004). A new feature selection score for multinomial naive bayes text classification based on kl-divergence. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, ACLdemo '04, Stroudsburg, PA, USA. Association for Computational Linguistics.

Wu, Q., Burges, C. J., Svore, K. M., and Gao, J. (2010). Adapting boosting for information retrieval measures. *Inf. Retr.*, 13(3):254–270.