Discretization Based Learning Approach to Information Retrieval

Dmitri Roussinov
Department of Information Systems
Arizona State University, Tempe, AZ, 85287
1-480-965-8488
dmitri.roussinov@asu.edu

Weiguo Fan
Department of Information Systems, Virginia Tech
3007 Pamplin Hall, Blacksburg, VA 24061
1-540-231-6588
wfan@vt.edu

Fernando A. Das Neves
Department of Computer Science
Virginia Tech
3007 Pamplin Hall, Blacksburg, VA 24061
1-540-231-6588
fdasneve@vt.edu

ABSTRACT

We have designed a representation scheme, which is based on the *discrete* representation of a document ranking function, which is capable of reproducing and enhancing the properties of such popular ranking functions as *tf.idf*, *BM25* or those based on language models. Our tests have demonstrated the capability of our approach to *achieve the performance of the best known scoring functions* solely through training, without using any known heuristic or analytic formulas.

Categories & Subject Descriptors: H.3.3

[**Information Storage And Retrieval**]: Information Search and Retrieval – *query formulation, search process*.

General Terms: Algorithms, Experimentation.

1. INTRODUCTION

Our work is motivated by the objective to bring closer numerous achievements in the domains of machine learning and classification to the classical task of ad-hoc information retrieval (IR), which is ordering documents by the estimated degree of relevance to a given query. Although used with striking success for text categorization, classification-based approaches have been relatively abandoned when trying to improve ad hoc retrieval in favor of empirical (e.g. vector space, bm25) or generative (e.g. language models). An important advantage of a discriminative approach is its ability to explicitly utilize the relevance judgments existing with standard test collections in order to train the IR algorithms and possibly enhance retrieval accuracy for the new (unseen) queries.

The review of the earlier (unsuccessful) attempts to involve discrimination based apporaches and more promising recent ones with larger (TREC) collections can be found in an extended version of this paper [3]. The major difference of our work from the prior is that *we did not try to combine several known ranking functions (or their separate terms) into one, but rather we learn the ranking functions directly through discretization.* The next section formalizes our Discretization Based Learning (DBL) approach to Information Retrieval, followed by empirical results and conclusions.

2. FORMALIZATION OF OUR APPROACH

We limit our ranking functions to the so called *lw.gw class*:

$$R(q, d) = \sum_{t \subset q} L(tf(t, d), d) G(t),$$

where d - document, q - query, *L*, local weighting, is the function of the number of occurrences of the term in the document *tf*, possibly combined with the other document statistics, e.g. word length. *G(t)*, global weighting, can be any collection level statistic of the term. We discretized the shape of the G*(t)* function by assigning each term to its global weighting bin *g*, which is an integer number in the *[1, |B|]* range, *|B|* is the total number of global weighting bins. The assignment of the term *t* to its global weighting bin *g(t)* is performed on the log linear scale according to the document frequency *df* of the term:

$$g(t) = \{| B | (1 - \frac{\log(df(t))}{\log(N)})\} \quad (1)$$

where *N* is the total number of documents, {.} stands for rounding down to the nearest integer.

Similarly to the global weighting, we assigned each occurrence of a term to its local weighting bin *l*, but this time by simply capping *tf* at the total number of local weighting bins *|L|*:

$$l(tf(t, d), d) = min(tf(t, d), |L|) \quad (1a)$$

Each occurrence of a query term in a document corresponds to a local/global bin combination *(g, l)*. Each *(g,l)* combination determines a feature in a vector representing a document-query pair ***f****(d, q)* and is denoted below as *f(d, q) [g , l]* . The dimensionality of the feature space is *|L| x |B|*. A feature vector ***f****(d, q)* represents each document *d* with respect to query *q*. The value of each feature in the vector is just the number of the term occurrences assigned to the pair of bins *(g, l)*:

$$f(d, q)[g, l] = \sum_{t \subset q, g(t)=g, l(t,d)=l} 1 \quad (2)$$

Since our features capture local (tf) and global (df) term occurrence information, in order to represent a ranking function, we can simply use the dot product between the feature vector and the vector of learned optimal weights **w**:

$R(q, d) = \mathbf{w} * \boldsymbol{f}(d, q)$.

Ideally, the learning mechanism should assign higher weights to the more important bin combinations (e.g. multiple occurrence of a rare term) and low weights to the less important combinations (e.g. single occurrence of a common term). The exact learned values determine the optimal shape of global and local weighting.

We still can make the representation more powerful by considering the learned weights *w[g, l]* not the replacements but rather the adjustments to some other chosen global *G (t)* and local *L (t, d)* weighting functions:

$$f(d, q)[g, l] = \sum_{t \subset q, g(t)=g, l(tf(t,d),d)=l} L(t, d) G(t) \quad (2a)$$

We define the specific choice of global *G()* and local *L()* weighting functions as *starting ranking function (SRF)*. When all the bin weights *w[g, l]* are set to *1*, our ranking function is the same as its *SRF*. The learning process finds the optimal values for *w[g, l]* for the collection of training queries and their relevance judgments, thus adjusting the important shapes of the global and local weighting to achieve better accuracy. *SRF* can be chosen from one of the known to perform well ranking functions (e.g. *tf.idf* or *BM25* or based on language models) to take advantage of the fact that those formulas and their optimal parameters on the standard test collections are

known for the researchers. Alternatively, we can set *SRF* to the constant value (e.g. *1* in formula 2), thus not taking advantage of any of the prior empirical investigations and to see if our framework is able to learn reasonable (or even top-notch) performance purely from labeled examples. Below, we describe our experiments with each approach.

Since the score is linear with respect to the feature values, we can train the weights $\mathbf{w}$ as a linear classifier that predicts the preference relation between pairs of documents with respect to the given query. Document *d1* is more likely to be relevant (has a higher score) than document *d2* iff $\boldsymbol{f}(d1, q) * \boldsymbol{w} > \boldsymbol{f}(d1, q) * \boldsymbol{w}$.

We chose support vector machines (SVM) for training the classifier weights *w[g, l]* since they are known to work well with large numbers of features, ranging in our experiments from 8 to 512, depending on the number of bins. For our empirical tests, we used the SVMLight package freely available for academic research from Joachims [1]. For each selected (sampled) pair of documents (*dr, di*), such that *dr* is a relevant document and *di* is irrelevant, the classifier was presented with a positive example created from the vector of differences of features $\boldsymbol{f_p} = \boldsymbol{f}(q, dr) - \boldsymbol{f}(q, di)$, and also with the negative example as the inverse of it: $\boldsymbol{f_n} = \boldsymbol{f}(q, di) - \boldsymbol{f}(q, dr)$. This approach also balances positive and negative examples.

Since presenting all pairs to the training mechanism would be overwhelming, we performed pseudo-random sampling of documents by the following intuitive consideration. Since it is more efficient to present the classifier with the pairs from the documents that are likely to more strongly affect the performance metric (average precision), we first pre-ordered the retrieved documents by any of the reasonably well-performing scoring function (e.g. *tf.idf*) and limited the sample of documents to the top *1000*. Then, for each query, each known relevant document *dr* from that subset was selected and "paired" with a certain number of randomly selected irrelevant documents. This number was linearly decreasing with the position of the relevant document in the pre-order. Thus, the higher the document was positioned in the pre-order, the more times it was selected for pairing (training). This placed more emphasis at correctly classifying the more important document pairs in the average precision computation. Without the correct emphasis during sampling the obtained results were much weaker.

3. Empirical Evaluation

3.1 Empirical setup

We used the TREC, Disks 1 and 2, collections to test our framework. We used topics 101-150 for training and 151-200 for testing and vice-versa. For indexing, we used the Lemur package [2], with the default set of parameters, and no stop word removal or stemming. We used only topic titles for queries, and the most popular average (non-interpolated) precision as our performance metric, computed by the script included with the Lemur toolkit (later verified by trec_eval). We used the implementation of *BM25*, available in Lemur as the baseline. The optimal parameter values were close to the default $K = 1.0$ and $b = .5$.

First, we set our starting ranking function (SRF) to a constant value, thus using only the minimum out of the empirical knowledge and theoretical models developed by information retrieval researchers during several decades: specifically only the fact that relevance can be predicted by tf and df. Table 1 shows performance for the 16 x 8 combination of bins. It can be seen that our approach has reached 90-100% of the top performance (baseline) solely through the learning process. The *original* performance is the one obtained by assigning all the classifier weights to 1. In order to evaluate if more training data can help, we also ran tests using 90 topics for training and the remaining 10 for testing. We ran 10 tests each time using 10 different sequential topics for testing and averaged our results. In this case, *the averaged performance was completely restored to the baseline level* with the mean difference in precision across test queries +0.5% and 1% standard deviation of the mean.

In order to test whether our approach can exceed the baseline performance we set *BM25* to be our starting ranking function (SRF). Table 2 shows performance for the 8 by 8 bin design. Although the improvement is relatively small (2-3%) it is still statistically significant at the level of alpha < 0.1, when the paired t-test was performed. The value in "*% change*" column shows the mean % improvement across all the queries and its standard deviation.

4. CONCLUSIONS, LIMITATION AND FUTURE RESEARCH

We explored learning how to rank documents with respect to a given query using linear Support Vector Machines and discretization-based representation. Our approach represents a family of discriminative approaches, currently studied much less than heuristic (*tf.idf*, *bm25*) or generative approaches (language models). Our experiments indicate that *learning from relevant judgments available with the standard test collections and generalizing to new queries is not only feasible but can be a source of improvement.* Using only one set of topics sets is a limitation of this current study, which we are going to address in our future research.

5. ACKNOWLEDGEMENT

Weiguo Fan's work is supported by NSF under the grant number ITR0325579. Roussinov's work is supported by Dean's Award of Excellence, W.P. Carey School of Business, summer 2005.

References

[1] Joachims, T. (2001). A Statistical Learning Model of Text Classification with Support Vector Machines. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2001.

[2] Kraaij, W., Westerveld T. and Hiemstra, D. (2003). The Lemur Toolkit for Language Modeling and Information Retrieval, http://www-2.cs.cmu.edu/~lemur

[3] Roussinov, D., and Fan, W., Discretization Based Learning Approach to Information Retrieval. *In proceedings of 2005 Conference on Human Language Technologies* (to appear).

Testing:	**101-150**			**151-200**		
Training:	Original	Learned	Baseline	Original	Learned	Baseline
101-150	.119	.165	.174	.135	.180	.204
151-200	.119	.175	.174	.135	.206	.204

Table 1. Learning without any knowledge of ranking functions. 16 x 8 bin design.

Testing:	**101-150**			**151-200**		
Training:	Learned	Baseline	% change	Learned	Baseline	
101-150	.180	.174	+2.3 (+/- 0.9)	.208	.204	+2.3 (+/- 1.0)
151-200	.179	.174	+1.8 (+/- 1.0)	.210	.204	+3.2 (+/- 1.3)

Table 2. Surpassing the baseline performance. 8 x 8 bin design.