Discretization Based Learning Approach to Information Retrieval

Dmitri Roussinov
Department of Information Systems
W.P. Carey School of Business
Arizona State University

Tempe, AZ, 85287
dmitri.roussinov@asu.edu

Weiguo Fan
Department of Information Systems and Computer Science
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061
wfan@vt.edu

Abstract

We approached the problem as *learning how to order documents* by estimated relevance with respect to a user query. Our *support vector machines* based classifier learns from the relevance judgments available with the standard test collections and generalizes to *new, previously unseen queries*. For this, we have designed a representation scheme, which is based on the *discrete* representation of the local (*lw*) and global (*gw*) weighting functions, thus 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 with the standard test collections have demonstrated the capability of our approach to *achieve the performance of the best known scoring functions* solely from the labeled examples and without taking advantage of knowing those functions or their important properties or parameters.

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 (e.g. those based on support vector machines, Joachims, 2001) have been relatively abandoned when trying to improve ad hoc retrieval in favor of empirical (e.g. vector space, Salton & McGill, 1983) or generative (e.g. language models; Zhai & Lafferty 2001; Song & Croft; 1999), which produce a ranking function that gives each document a score, rather than trying to learn a classifier that would help to discriminate between relevant and irrelevant documents and order them accordingly. A generative model needs to make assumptions that the query and document words are sampled from the same underlying distributions and that the distributions have certain forms, which entail specific smoothing techniques (e.g. popular Dirichlet-prior). A discriminative (classifier-based) model, on the other side, does not need to make any assumptions about the forms of the underlying distributions or the criteria for the relevance but instead, learns to predict to which class a certain pattern (document) belongs to based on the labeled training examples. Thus, an important advantage of a discriminative approach for the information retrieval task, 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.

Cohen, Shapire and Singer (1999) noted the differences between ordering and classification and presented a two-stage model to learn ordering. The first stage learns a classifier for preference relations between objects using any suitable learning mechanism (e.g. support vector machines; Vapnik, 1998). The second stage converts preference relations into a rank order. Although the conversion may be NP complete in a general case, they presented efficient approximations. We limited our first study reported here to linear classifiers, in which conversion can be performed by simple ordering according to the score of each document. However, approaching the problem as "learning how to order things" allowed us to design our sampling and training mechanisms in a novel and, we believe, more powerful way.

Our classifier learns how to compare every pair of documents with respect to a given query, based on the relevance indicating features that the documents may have. As it is commonly done in information retrieval, the features are derived from the word overlap between the query and documents. According to Nallapati (2004), the earliest formulation of the classic IR problem as a classification (discrimination) problem was suggested by Robertson and Sparck Jones (1976), however performed well only when the relevance judgments were available for the same query but not generalizing well to new queries. Fuhr and Buckley (1991) used polynomial regression to estimate the coefficients in a linear scoring function combining such well-known features as a weighted term frequency, document length and query length. They tested their "description-oriented" approach on the standard small-scale collections (Cranfield, NPL, INSPEC, CISI, CACM) to achieve the relative change in the average precision ranging from -17%

Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP), pages 153–160, Vancouver, October 2005.

to + 33% depending on the collection tested and the implementation parameters. Gey (1994) applied logistic regression in a similar setting with the following results: Cranfield +12%, CACM +7.9%, CISI -4.4%, however he did not test them on new (unseen by the algorithm) queries, hypothesizing that splitting documents into training and testing collections would not be possible since "a large number of queries is necessary in order to train for a decent logistic regression approach to document retrieval." Instead, he applied a regression trained on Cranfield to CISI collection but with a negative effect.

Recently, the approaches based on learning have reported several important breakthroughs. Fan et al. (2004) applied genetic programming in order to learn how to combine various terms into the optimal ranking function that outperformed the popular Okapi formula on robust retrieval test collection. Nallapati (2004) made a strong argument in favor of discriminative models and trained an SVM-based classifier to combine 6 different components (terms) from the popular ranking functions (such as tf.idf and language models) to achieve better than the language model performance in 2 out of 16 test cases (figure 3 in Nallapati, 2004), not statistically distinguishable in 8 cases and only 80% of the best performance in 6 cases. There have been studies using past relevance judgements to optimize retrieval. For example, Joachims (2002) applied Support Vector Machines to learn linear ranking function from user click-throughs while interfacing with a search engine.

In this study, we have developed a representation scheme, which is based on the discretization of the *global* (corpus statistics) and *local* (document statistics) weighting of term overlaps between queries and documents. We have empirically shown that this representation is flexible enough to learn the properties of the popular ranking functions: tf.idf, BM25 and the language models. The major difference of our work from Fan et al. (2004) or Nallapati (2004) or works on fusion (e.g. Vogt & Cottrell, 1999) is that *we did not try to combine several known ranking functions (or their separate terms) into one, but rather we learn the weighting functions directly through discretization.*

Discretization allows representing a continuous function by a set of values at certain points. These values are learned by a machine learning technique to optimize certain criteria, e.g. average precision.

Another important motivation behind using *discretization* was to design a representation with high dimensionality of features in order to combine our representation scheme with Support Vector Machines (SVM) (Vapnik, 1998), which are known to work well with a large number of features. SVM contains a large class of neural nets, radial margin separation (RBF) nets, and polynomial classifiers as special cases. They have been delivering superior performance in classification tasks in general domains, e.g. in face recognition (Hearst, 1998), and in text categorization (Joachims, 2001).

Another important distinction of this work from the prior research is that we train our classifier not to predict the absolute relevance of a document *d* with respect to a query *q*, but rather to predict which of the two documents *d1*, *d2* is more relevant to the query *q*. The motivation for this distinction was that all the popular evaluation metrics in information retrieval (e.g. average precision) are based on document ranking rather than classification accuracy. This affected our specially designed sampling procedure which we empirically discovered to be crucial for successful learning.

We have also empirically established that our combination of the representation scheme, learning mechanism and sampling allows learning from the past relevance judgments in order to successfully generalize to the new (unseen) queries. When the representation was created without any knowledge of the top ranking functions and their parameters, *our approach reached the known top performance solely through the learning process*. When our representation was taking advantage of functions that are known to perform well and their parameters, the resulting combination was *able to slightly exceed the top performance on large test collections*. The next section formalizes our Discretization Based Learning (DBL) approach to Information Retrieval, followed by empirical results and conclusions.

2. Formalization Of Our Approach

2.1 Query and Document Representation

We limit our representation to the so called *lw.gw class*: $R(q, d) = \sum_{t \subset q} L(tf(t, d), d) G(t)$,

where *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. For example, in the classical *tf.idf* formula $L(tf, d) = tf / |d|$, where *tf* is the number of occurrences of the term *t* in the document, $|d|$ is the length of the document vector and $G(t) = \log (N / df(t))$, where *df(t)* is the total number of documents in the collection that have term *t* and *N* is the total number of documents.

Without loss of generality it may also be extended to handle a number of occurrences of the term in the query, but we omit it here in our formalization for simplicity. *Lw.gw* class includes the BM25 Okapi ranking function which performs well on TREC collections (Robertson et al., 1996). It can be shown that many of the recently introduced language models fall into that category as well, specifically the best performing in TREC ad hoc tests Dirichlet smoothing, Jelinek Mercer smoothing, and Absolute Discounting approaches can be represented that way (see equation 6 and table I in Zhai & Lafferty, 2001). An *lw.gw* representation of Jelinek Mercer smoothing was used in Nallapati (2004). It has been known for a long time that the shapes of the global and local weighting functions can *dramatically affect the precision* in standard test collections because it in

fact determines the difference between such formulas as *tf.idf, bm25* and language models. *However, we are not aware of any attempts to learn those shapes directly from the labeled examples, which we performed in this study.*

2.2 Intuition behind the discretization-based learning

The intuition behind discretization approach is to represent a function by values at the finite number of points. Then, the optimal shape of the function can be learned by using one of the machine learning techniques. Our discretization based learning (DBL) approach to information retrieval learns how important each class of an occurrence of a query term in a document. For example, in some very "primitive" DBL approach, we can define two classes: Class S ("strong"), containing all multiple occurrences of a rare query term (e.g. "discretization") in a document and Class W ("weak"), containing all single occurrences of a frequent term (e.g. "information"). Then, the machine learning technique should discover that the occurrences of Class S are much stronger indicators of relevance than the occurrences of Class W. In the DBL implementation presented in this paper, each occurrence of a query term is assigned to a class (called *bin*) based on the term document frequency in the collection (*df*) and the number of occurrences within the document (*tf*). The bin determines the weight of the contribution of each occurrence of the query term in the ranking score. Thus, the relevance score is just the weighted sum of the numbers of occurrences within each bin. The other way of looking at it is that the score is produced by a linear classifier, where the total number of occurrences within each bin serves as the feature value. *By learning the optimal weights, a linear classifier effectively learns the optimal shapes of the global (gw) and local (lw) weighting functions. By learning the discrimination properties of each bin, rather than separate word terms, DBL method allows generalization to new queries.*

2.3 Discretizing global weighting

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:

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

where *N* is the total number of documents, {.} stands for rounding down to the nearest integer. The logarithmic scale allows more even term distribution among bins than simple linear assignment, which is desirable for more efficient learning. It is motivated by a typical histogram of *df(t)* distribution, which looks much more uniform in a logarithmic scale. It is important to note that it does not have anything to do with the *log* function in the classical *idf* weighting and that the formula for *g(t)* does not produce any weights but only assigns each term occurrence to a specific bin based on the term document frequency. The weights are later trained and effectively define any shape of global weighting, including such simple functions tried in the prior heuristic explorations as logarithm, square root, reciprocal and others.

2.4 Discretizing local weighting

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|*:

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

Let's note that this particular representation does not really need rounding since *tf* is already a positive integer. However, in a more general case, *tf* can be normalized by document length (as is done in BM25 and language models) and thus local weighting would become a continuous function. It is important to note that our discrete representation does not ignore the occurrences above *|L|* but simply treats them the same way as *tf = |L|*. The intuition behind capping is that increasing *tf* above certain value (|L|) would not typically indicate the higher relevance of the document. Typically, a certain number of occurrences is enough to indicate the presence of the relevant passage. Please note again that this bin assignment does not assign any heuristic weights to the term occurrences.

2.5 Final discretized ranking function

The bin assignments based on *tf* and *df* specified in sections 2.3 and 2.4 are straightforward and do not involve any significant "feature engineering." 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|*. E.g. for 8 local weighting bins and 10 global weighting bins we would deal with the vector size of 80. 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):*

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

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) = **w** * ***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:

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

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 **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 $\boldsymbol{f}(d1, q) * \boldsymbol{w} > \boldsymbol{f}(d1, q) * \boldsymbol{w}$. An important advantage of using a linear classifier is that rank ordering of documents according to the learned pairwise preferences can be simply performed by ordering according to the linear score. Please refer to Cohen et al. (1999) for the ordering algorithms in a more general non linear case.

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 (2001). We preserved the default parameters coming with version V6.01. Although SVMLight package allows learning ranking, we opted for training it as a classifier to retain more control over sampling, which we found crucial for successful learning, as described in the section below.

2.6 Sampling

Since we were training a classifier to predict preference relations, but not the absolute value of relevance, we trained on the differences between feature vectors. Thus, 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.

We also informally experimented with training on absolute relevance judgments, similar to the prior work mentioned in the Introduction but obtained much worse results. We explain it by the fact that relative judgments (pairwise comparisons) are more generalizable to new queries than absolute judgments (relevant/irrelevant). This may explain prior difficulties with applying discriminative approaches mentioned in our Introduction.

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. Again, without the correct emphasis during sampling the obtained results were much weaker. However, the choice of the ranking function to perform pre-order was found to be not important: virtually the same results were obtained using *tf.idf* or bm25 or language models.

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 (Kraaij et al., 2003), with the default set of parameters, and no stop word removal or stemming. Although those procedures are generally beneficial for accuracy, it is also known that they do not significantly interfere with testing various ranking functions and thus are omitted in many studies to allow easier replication.

We used only topic titles for queries, as it is commonly done in experiments, e.g. in Nallapati (2004). We used 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). The characteristics of the collection after indexing are shown in Table 1. We also reproduced results similar to the reported below on the Disk 3 collection and

topics 101-150, but did not include them in this paper due to size limitations.

Collection	TREC Disks 1 and 2
Number of documents	741,863
Number of terms	325,059,876
Number of unique terms	697,610
Average doc. length	438
Topics	101-200

Table 1. The characteristics of the test collection: TREC Disks 1,2

3.2 The baseline

In this study, we were interested exclusively in the improvements due to learning, thus still staying within the "bag of words" paradigm. Although many enhancements can be easily combined within our framework, we limited our search for the baseline performance to "bag of words" techniques to avoid unfair comparison. We used the results reported in Nallapati (2004) as guidance and verified that the best performing language model on this test collection was the one based on the Dirichlet smoothing with $\mu = 1900$. Our average precision was lower (*0.205* vs. *0.256*), most likely due to the different indexing parameters, stemming or using a different stopword list. By experimenting with the other ranking functions and their parameters, we noticed that the implementation of *BM25*, available in Lemur, provided almost identical performance (*0.204*). Its ranking function is $BM25\ (tf, df) = tf / (tf + K^* (1 - b + b * |d| / |d|_a) * log\ (N / (df + .5))$, where $|d|$ is the document word length and $|d|_a$ is its average across all documents. The optimal parameter values were close to the default $K = 1.0$ and $b = .5$. We noticed that the query term frequency components could be ignored without any noticeable loss of precision. This may be because the TREC topic titles are short and the words are very rarely repeated in the queries. Since the difference between this ranking function and the optimal from the available language models was negligible we selected the former as both our baseline and also as the starting ranking function (SRF) in our experiments. For simplicity, we call it simply *BM25* throughout our paper.

3.3 Discretization accuracy

Before testing the learning mechanism, we verified that the loss due to discretization is minimal and thus the approach is capable of capturing global and local weighting. For this, we discretized our baseline BM25 formula replacing each score contribution of the occurrence of a term $G(t)L(t,d) = BM25(t, d)$ with its average across all other occurrences within the same bin combination $[g, l]$, which is determined by the formulas 1 and 1a. We discovered that for the $|B| \times |L| = 8 \times 8$ configuration, the loss in average precision did not exceed *2%* (relatively). This demonstrates that the $G(t)L(t,d)$ ranking functions can be discretized (replaced by values at certain points) at this level of granularity without losing much accuracy. We also verified that the weights $w[g, l]$ can affect the performance significantly: when we set them to random numbers in the [0,1] range, the performance dropped by 50% relatively to the baseline.

3.4 Ability to achieve top performance from scratch

First, we were curious to see if our framework can learn reasonable performance without taking advantage of our knowledge of the top ranking functions and their parameters. For this, 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 2 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. It can be seen that the topics 151-200 are more amenable for the technique that is why they show better recovery when used as a test set even when the training set 101-150 recovers only 90%. 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.

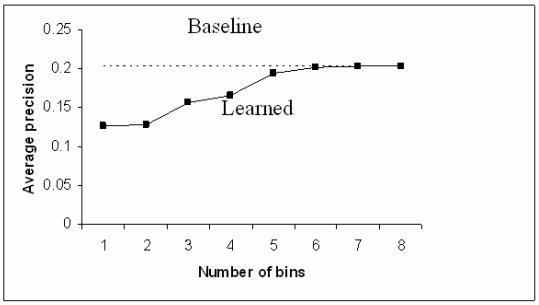

Figure 1. Learning local weighting for various

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

numbers of bins. Learning on 101-150 and testing on 151-200.

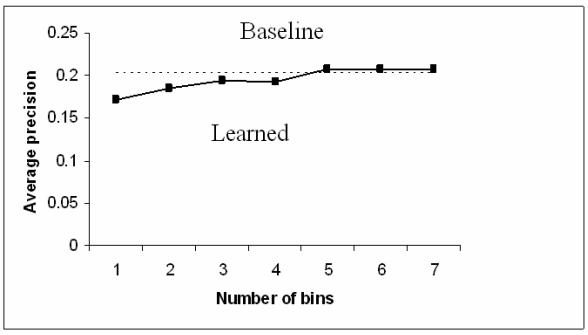

Figure 2. Learning global weighting for various numbers of bins. Learning on 101-150 and testing on 151-200.

We believe this is a remarkable result considering the difficulties that the prior learning based approaches had with the classical information retrieval task. We attribute our success to both higher flexibility and generalizability of our discrete representation. We also varied the number of bins to evaluate the effect of granularity of representation. Figures 1 and 2 demonstrate that 8 bins suffice for both global and local weighting. Higher numbers did not result in noticeable improvements.

When the same set was used for training and testing the result obviously overestimates the learning capability of the framework. However, it also gives the upper bound of performance of a discretized *gw.lw* combination assuming that the loss due to discretization is negligible which can be easily attained by using sufficiently large number of bins. *Thus, the results indicate that gw.lw, which includes practically all the popular "bag of words" ranking formulas such as tf.idf, BM25 or language models, has almost reached its upper limit and other classes of representations and ranking formulas need to be explored to attempt greater improvements.*

Figure 2. Learning global weighting for various numbers of bins. Learning on 101-150 and testing on 151-200.

3.5 Ability to surpass top performance

In order to test whether our approach can exceed the baseline performance we set *BM25* to be our starting ranking function (SRF). Thus, in this case:

$$G(t) = \log (N / (df + .5)) \quad (6)$$

$$L(tf, d) = tf / (tf + K * (1 - b + b * |d| / |d|_a)$$

Table 3 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. It may differ from the % change of the mean performance since there is wide variability in the performance across queries but smaller variability in the improvement.

We believe even such a small improvement is remarkable considering the amount of attention the researches have paid to optimizing the ranking functions for this specific data set which has been available for more than seven years. A number of recent studies reported comparable improvements on the same test collection by using more elaborate modeling or richer representations. Of course the improvement due to the techniques such as those based on n-grams, document structures, natural language processing or query expansion can possibly achieve even better results. However in this study we deliberately limited our focus to the "bags of words."

3.6 Shape of optimal local weighting

Figure 3 shows the optimal shape of the local weighting function *L(tf)* learned on entire set of 100 topics and plotted against their counterparts of *BM25(t, d) = tf / (tf + 1)* and *tf.idf(t, d) = tf* for comparison. For plotting purposes, we assumed that the document length was equal to its average. The values were linearly scaled to meet at the *tf = 8* point. It is easy to observe that the behavior of the optimal function is much closer to BM25 than to *tf.idf*, which explains the good performance of the former on this test set.

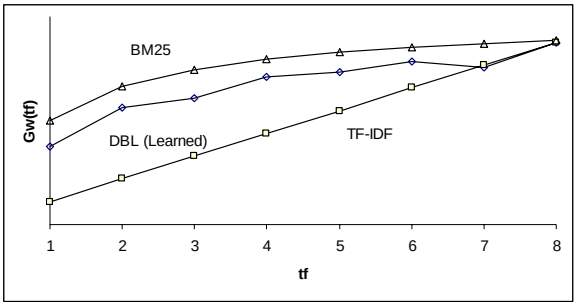

Figure 3. Learned optimal shape of local weighting.

3.7 Shape of optimal global weighting

Figure 4 shows the optimal shape of the global weighting function *G(t)* learned on the entire set of 100 topics with *|B| = 32* plotted in logarithmic scale against the popular *idf* weighting used in both *tf.idf* and BM25. The lower end of the X-axis (*log10 df < 2*) corresponds to very infrequent terms, so the learned weights may not be very informative since

the classifier encounters fewer occurrences of them and their impact on the overall accuracy is small. In the mid range (5,000 – 10,000), the optimal weights are higher than idf, which indicates that the latter has an overly steep shape to discount high frequency terms. A more detailed interpretation of the optimal shape may require further investigation.

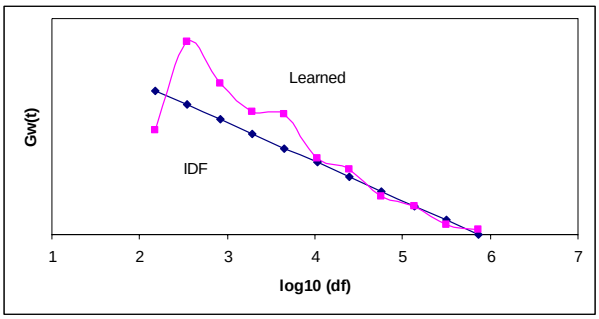

Figure 4. Learned optimal shape of global weighting G(t).

4. Conclusions

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.* When tested with a popular standard collection, our approach achieved the performance of the best well-known techniques (BM25 and language models), which have been developed as a result of extensive past experiments and elaborate theoretical modeling. When combined with the best performing ranking functions, our approach added a small (2-3%), but statistically significant, improvement.

Although practical significance of this study may be limited at the moment since it does not demonstrate a dramatic increase in retrieval performance in large test collections, we believe our findings have important theoretical contributions since they indicate that the power of discriminative approach is comparable to the best known analytical or heuristic apporaches. This work also lays the foundation for extending the discriminative approach to "richer" representations, such as those using word n-grams, grammatical relations between words, and the structure of documents.

Our results also indicate that *gw.lw* family, which includes practically all the popular "bag of words" ranking formulas such as tf.idf, BM25 or language models, has almost reached its upper limit and other classes of representations and ranking formulas need to be explored in order to accomplish significant performance break-troughs.

Of course, using only few test cases (topics sets and collections) is a limitation of this current study, which we are going to address in our future research. We view our approach as a complement, rather than competitive, to the analytical approaches such as language models. Our approach can be also used as an explorative tool in order to identify important relevance-indicating features, which can be later modeled analytically. We believe that our work and the ones referred in this paper may bring many of the achievements made in a more general area of classification and machine learning closer to the task of rank ordered information retrieval, thus making retrieval engines more helpful in reducing the information overload and meeting people's needs.

5. Acknowledgement

Weiguo Fan's work is supported by NSF under the grant number ITR0325579.

References

Bartell, B., Cottrell, G., and Belew, R.(1994). Optimizing Parameters in a Ranked Retrieval System Using Multi-Query Relevance Feedback. *Symposium on Document Analysis and Information Retrieval (SDAIR).*

Chengxiang Zhai and John Lafferty (2001). A study of smoothing methods for language models applied to Ad Hoc information retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, pp. 334 – 342, 2001.

Cohen, W., Shapire, R., and Singer, Y. (1999). Learning to order things. Journal of Artificial Intelligence Research, 10, 243-270, 1999.

Dougherty, J., Kohavi, R., & Sahami, M. (1995). Supervised and unsupervised discretization of continuous features. *Proceedings of the Twelfth International Conference on Machine Learning* (pp. 194--202). Tahoe City, CA: Morgan Kaufmann.

Fan, W., Luo, M., Wang, L., Xi, W., and Fox, A. (2004). Tuning Before Feedback: Combining Ranking Discovery and Blind Feedback for Robust Retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2004.

Fuhr, N. and C. Buckley (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9*, 223–248.

Fuhr, N. and C. Buckley (1991). A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems, 9*, 223–248.

Gey, F. C. (1994). Inferring probability of relevance using the method of logistic regression. In *Proceedings of the 17th ACM*

Conference on Research and Development in Information Retrieval (SIGIR'94), pp. 222–231.

Hearst, M. (1998). Support Vector Machines. *IEEE Intelligent Systems Magazine, Trends and Controversies*, Marti Hearst, ed., 13(4), July/August 1998.

Hun-Nan Hsu, Hung-Ju Huang and Tzu-Tsung Wong (2000). Why Discretization Works for Naive Bayesian Classifiers, In Proceedings of the 17th International Conference on Machine Learning (ICML-2000), Stanford, CA, USA. Page 399-406.

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.

Joachims, T. (2002). Optimizing Search Engines Using Clickthrough Data, *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining (KDD)*, ACM, 2002.

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

Nallapati, R. (2004). Discriminative models for information retrieval. *Proceedings of the Conference on Research and Development in Information Retrieval (SIGIR)*, 2004, pp. 64-71.

Robertson S. E. and Sparck Jones, K. (1976). Relevance weighting of search terms, *Journal of American Society for Information Sciences*, 27(3), pp. 129-146, 1976.

Robertson, S. E., Walker, S., Jones S., Hancock-Beaulieu M.M., and Gatford, M. (1996)., Okapi at TREC-4, in D. K. Harman, editor, Proceedings of the Fourth Text Retrieval Conference, pp. 73–97. NIST Special Publication 500-236, 1996.

Salton, G. and McGill, M.J. (1983). Introduction to Modern Information Retrieval. New York. McGraw-Hill.

Song, F. and W.B. Croft. (1999) A general language model for information retrieval. *In Proceedings of Eighth International Conference on Information and Knowledge Management* (CIKM'99).

Vapnik, V. N. (1998).. Statistical Learning Theory. John Wiley and Sons Inc., New York, 1998.

Vogt, C., G. Cottrell, G. (1999). Fusion Via a Linear Combination of Scores. *Information Retrieval*, 1(3), pp. 151—173.