On Ranking the Effectiveness of Searches

Vishwa Vinay, Ingemar J. Cox
University College London, London, UK
v.vinay@cs.ucl.ac.uk, ingemar@ieee.org

Natasa Milic-Frayling, Ken Wood
Microsoft Research Ltd., Cambridge, UK
natasamf@microsoft.com, krw@microsoft.com

ABSTRACT
There is a growing interest in estimating the effectiveness of search. Two approaches are typically considered: examining the search queries and examining the retrieved document sets. In this paper, we take the latter approach. We use four measures to characterize the retrieved document sets and estimate the quality of search. These measures are (i) the clustering tendency as measured by the Cox-Lewis statistic, (ii) the sensitivity to document perturbation, (iii) the sensitivity to query perturbation and (iv) the local intrinsic dimensionality. We present experimental results for the task of ranking 200 queries according to the search effectiveness over the TREC (discs 4 and 5) dataset. Our ranking of queries is compared with the ranking based on the average precision using the Kendall τ statistic. The best individual estimator is the sensitivity to document perturbation and yields Kendall τ of 0.521. When combined with the clustering tendency based on the Cox-Lewis statistic and the query perturbation measure, it results in Kendall τ of 0.562 which to our knowledge is the highest correlation with the average precision reported to date.

Categories and Subject Descriptors
H.3.3 Information Search and Retrieval

General Terms
Algorithms

Keywords
Query Performance Prediction

1. INTRODUCTION
There is a considerable interest within the Information Retrieval community in estimating the effectiveness of search. Having such a measure would be useful for a variety of purposes identified in [1] and include (i) providing feedback to the user, (ii) providing feedback to the search engine, (iii) providing feedback to the database creators, and (iv) optimizing information fusion for meta-search engines.

A number of strategies have recently been proposed for estimating search effectiveness. They can be broadly categorized into two classes. The first class is based on an analysis of the query. Cronen-Townsend *et al* [2] define a *clarity score* that depends on a language model. Queries which fit the language model of the entire document collection are considered too general, leading to a low clarity score. In contrast, a query that identifies only a subset of the collection has a high specificity and thus a high clarity score.

Amati *et al* [3], on the other hand, propose a "divergence from randomness" framework that uses an information theoretic function to predict the average precision for a given query. This quantity is then used to apply query expansion selectively. He and Ounis [4] describe a method for predicting query effectiveness based on the query length and the distribution of the inverse document frequency values for each of the constituent terms. Unfortunately, these methods were able to achieve only limited success, indicating the difficulty of the task and the need for more elaborate methodologies.

The second class of algorithms is based on an analysis of the retrieved document set. Yom-Tov *et al* [1] propose a method that uses a variety of heuristic features, including the overlap of retrieval sets obtained using the query and each of its sub-queries. It is assumed that the larger the agreement across retrieval sets, the more likely it is that a suitable set of documents has been retrieved. Other features include the score of the highest ranking document and the number of words in the query. The features are linearly combined using weights that are estimated from a learning phase that requires a set of ranked query/response data as a training set. After training, the experimental results reported in [1] demonstrated the best performance to date, with a Kendall τ statistic of 0.439 using the same dataset as we used in the work reported here.

In this paper, we propose four measures for estimating query performance that focus on the geometry of the retrieved document set: (i) the clustering tendency as measured by the Cox-Lewis statistic, (ii) the sensitivity to document perturbation, (iii) the sensitivity to query perturbation, and (iv) the local intrinsic dimensionality. Sections 2-5 provide a detailed description of each. Our previous work reported the use of the clustering tendency alone to estimate search effectiveness. Here we extend that work and significantly improve the performance of our predictor. Section 6 reports experimental results for ranking 200 queries based on their search effectiveness over the TREC discs 4 and 5 dataset. The relevant documents for these queries are known and used to calculate the average precision for each query. The precision statistics provides the ground-truth ranking of queries that is then used for comparison with other methods and calculating the corresponding Kendall τ statistic. Section 7 provides a summary and discussion.

SIGIR'06, August 6–11, 2006, Seattle, Washington, USA.

2. MEASURING THE CLUSTERING TENDENCY

The "cluster hypothesis" [5] states that documents relevant to a given query are likely to be similar to each other. Thus, documents relevant to a query are expected to form a group that is distinct from non-relevant documents. In practice, we attempt to exploit this hypothesis in its equivalent form: we expect the lack of clusters in the retrieved data sets to imply that the set does not contain relevant documents, provided that the set is large enough, i.e., larger than the expected number of relevant documents for the query. In other words, detecting a high level of 'randomness' in the retrieved set implies the absence of relevant documents and thus low precision for the given query.

We find a considerable body of literature in pattern learning that covers the clustering tendency of a set of points. A good introduction to useful techniques can be found in [6]. For our purposes we use a modified version of the Cox-Lewis statistic defined in [7].

The Cox-Lewis statistic is based on the ratio of two distances: (i) the distance from a randomly generated sampling point to its nearest neighbor in the dataset, called the *marked point* and (ii) the distance between the marked point and its nearest neighbor. A rigorous calculation of this statistic requires that we define a spatial point process which models the generation of the data and provides the initial random points. We apply a much simplified version where instead of using a spatial random process, we pick points from within the dataset and replace their weights by random values chosen from within a *sampling window* and these serve as our random origins. This is explained in more detail in Section 2.1 below.

When the data contains inherent clusters, the distance D_{rand} between the random sampling point and its marked point, i.e., the closest neighbor in the dataset, is likely to be much larger than the distance D_{nn} between the marked point and its nearest neighbor. This quantity D_{rand}/D_{nn} should therefore reflect the presence or absence of inherent groupings present in the data.

2.1 Approximation of the Cox-Lewis statistic

As has been noted in previous literature [6], the definition of the *sampling window,* a region in the data representation space from which the 'random' points are picked, is an important factor for the Cox-Lewis statistic. We are trying to measure a property that is *internal* to the data and thus the random points from the data set need to be chosen with care. In our case we define the sampling window to be the smallest hyper-rectangle that contains all the documents in the retrieved set. Since the size of this hyper-rectangle can vary significantly between queries, we normalize the computed Cox-Lewis ratio by the average length of the sides of the hyper-rectangle.

Once the sampling window has been identified, we generate each random point by starting with a point randomly selected from the retrieved set of documents. We replace each non-zero term weight with a value chosen uniformly from the range that corresponds to the side of the hyper-rectangle in that dimension. The points from the dataset are thus used only to determine which components to assign a random value to.

The clustering tendency of a given set of points, in our case the retrieved documents, is dependant on their sparsity, i.e., the proportion of non-zero values in the matrix representation of the points. The sampling points serve as pseudo-data points and therefore replacing only the non-zero components by randomly chosen values maintains the dependency on the sparsity while eliminating the need for defining a generative model of the data.

Having obtained a random sample point, we determine its nearest neighbor within the retrieved set, the *marked point*, and compute the distance between them. We then calculate the distance between the marked point and its nearest neighbor. The ratio of these two distances gives us an estimate of the randomness present within the retrieved set of documents.

When working with text documents we compute a cosine similarity between pairs of documents or a document and a query. For the purpose of query performance prediction, we further tailor this approach using Tombros and van Rijsbergen query-dependant extension of the dot-product. Amongst the alternatives suggested by Tombros and van Rijsbergen in [8], we calculate the similarity between two documents as the product of their cosine dot product and the query-dependant component:

$$Sim_{query}(\mathbf{d}_i, \mathbf{d}_j \mid \mathbf{q}) = \frac{\sum_{k=1}^{T} d_{ik} d_{jk}}{\sqrt{\sum_{k=1}^{T} d_{ik}^2 \sum_{k=1}^{T} d_{jk}^2}} * \frac{\sum_{k=1}^{T} c_k q_k}{\sqrt{\sum_{k=1}^{T} c_k^2 \sum_{k=1}^{T} q_k^2}}$$

where $\mathbf{d}_i$ and $\mathbf{d}_j$ are the two documents, T is the number of unique terms in the collection, $\mathbf{q}$ is the query (with weight q_k for term k) and $\mathbf{c}$ is the vector of terms common to both d_i and d_j with weights c_k being the average of d_{ik} and d_{jk}.

Thus, pairs of documents are close to each other if they share terms among themselves and with the query. Therefore, their distance is not an absolute value but relative to the search context, i.e., the query. If two documents do not contain query terms their query-dependant similarity will be 0 regardless of how close they may be with regards to the cosine similarity. We use this measure to determine both the marked points and their nearest neighbors in the dataset and thus obtain a *query specific Cox-Lewis statistic*. A higher clustering tendency is thus implicated by a larger value of the ratio:

$$CT_q = Mean\left(\frac{Sim_{query}(\mathbf{d}_{mp}, \mathbf{d}_{nn} \mid \mathbf{q})}{Sim_{query}(\mathbf{p}_{sp}, \mathbf{d}_{mp} \mid \mathbf{q})}\right) * \frac{1}{T}\sum_{i=1}^{T}(x_i - y_i)$$

where $\mathbf{q}$ is the query, $\mathbf{p}_{sp}$ is the sampling point, $\mathbf{d}_{mp}$ is the marked point, i.e., the document with largest similarity with the sampling point, and $\mathbf{d}_{nn}$ is the nearest neighbor of the marked point. Here x_i represents the maximum and y_i the minimum weight for a term i across the retrieved set. Both x_i and y_i are calculated when defining the sampling window. The quantity given in the above equation is taken to be proportional to the average precision of this query.

3. DOCUMENT PERTURBATION

Given a set of retrieved documents, consider the situation in which a document is randomly selected from the retrieved set and used as a pseudo-query over the retrieved data set. We expect the new result list to have that very document ranked first. Now let us examine what effect adding noise to the representation of such document would have. We issue a perturbed version of the document as a pseudo-query and record

```
For each query
{
    Issue query to dataset
    Collect 100 results
    Calculate the variance along each dimension from amongst documents that contain this term
    For each document d_i in this set
    {
        For alpha = 0.01, 0.1, 1, 10
        {
            For s = 1 : 10
            {
              For each term j present in this document
                  Weight = Original_weight + Gaussian(0, alpha* v_j)
              Find similarity of the noisy doc with all 100 original documents
              Find rank of d_i in the list of similarities
            }
            Find average rank over multiple samples for doc d_i and this alpha
        }
    }
    Find average rank across all 100 documents for each given alpha
    Plot average rank Vs alpha for the range of alphas
    Find slope of the line for this query
}
```

Figure 1: Pseudo-code for the document perturbation measure

the new rank that the original document assumes with respect to the search with the modified pseudo-query.

We perform this analysis for all of the documents in the retrieved set and calculate the rate at which the average rank changes depending on the noise. More precisely, we plot the increase in the document rank, (as the original document falls down the list), averaged over all the documents, against the level of introduced noise. We use the slope of the corresponding curve to estimate the retrieval performance for a query.

Specifically, consider a document $\mathbf{d}_i$ from the retrieved set containing *N* results for a query. Let **S** be the matrix that comprises *N* columns corresponding to the vectors of retrieved documents. When using the cosine dot product for similarities between documents, $\text{sim}(\mathbf{d}_i, \mathbf{d}_i) = \mathbf{d}_i \,.\, \mathbf{d}_i^T = 1$ where X^T denotes the transpose, we find that the i^{th} entry in the vector $<\mathbf{d}_i.\mathbf{S}^T>$ is 1 and represents the maximum similarity value in the resulting vector, assuming that the retrieved set does not contain duplicate documents. We add noise to the document vector $\mathbf{d}_i$ as follows. Every non-zero term-weight d_{ij} of this document is altered by adding to it a random value drawn from a Gaussian with 0-mean and variance $\alpha * v_j$. The value v_j is the variance for term j seen across **S**. Increasing α increases the magnitude of the noise.

The perturbed document $\mathbf{d}_i'$ differs from $\mathbf{d}_i$ and therefore it will not have a similarity of 1 when compared with the unperturbed version $\mathbf{d}_i$. We count the number of elements in $<\mathbf{d}_i'.\mathbf{S}^T>$ that are larger than the product $\mathbf{d}_i'.\mathbf{d}_i^T$ to determine the new rank of the original document. As α increases, the amount of noise increases and the rank continues to fall before stabilizing at *N*/2, which is essentially a random ranking.

It has to be noted that if a fixed amount of noise is added to a random set of points and a clustered set of points, respectively, the clustered set is likely to be more prone to a fast change in rank since the points are tightly grouped. However, in our case, the noise added is data dependant. For a given value of α, the magnitude of introduced noise is dependant on the variance observed in the given data set. Between a random and a clustered set of points, the random set is likely to have a larger variance and therefore have a larger noise added for the same α. This, in turns, leads to a larger change in rank of the original document when the perturbed document is used as a query. It is, therefore reasonable to expect that the inverse of the rate of change of document ranks with $\log(\alpha)$ will be related to the clustering properties and, consequently, to the average precision of the query.

In our experiments, we selected the range of α through experimentation, aiming at the amount of noise that is sufficient to induce a regular and monotonic behavior but not too high to causes erratic behavior. Since the perturbation process involves an element of randomness, we resort to multiple samples averaging to ensure the stability of the observed measure. The rank of a document at a given level of noise(α) is calculated as an average over ten samples. The pseudo-code for this process is given in Figure 1.

4. QUERY PERTURBATION

For a specific query and a retrieval model, the terms within the query are given particular weights. Altering these weights by a controlled addition of noise produces a perturbed, "noisy", query. If the retrieval algorithm is a nearest neighbor search, the set of documents retrieved by the original query will most likely be different from the set retrieved by the perturbed query. Here, we attempt to measure how distant the original retrieved set is from the documents in the collection that would be retrieved as a result of small perturbations in the query.

Our rationale is that if the originally retrieved set of documents forms a tight cluster that is significantly distant from other topical clusters in the collection, and if the magnitude of the

```
For each query
{
        Issue query to dataset
        Collect 100 results – called the original_set
        Calculate the variance along each dimension from amongst documents that contain this term
        For alpha = 0.01, 0.1, 1, 10
        {
                For s = 1 : 10
                {
                   For each term k present in this query
                       Weight = Original_weight + Gaussian(0, alpha* v_k)
                  Issue query to the entire dataset
                  Collect 100 results – called the noisy_set
                  Find the Levenshtein distance between original_set and noisy_set
                }
                Find average distance over multiple samples for this alpha
        }
        Find average distance for each given alpha
        Plot average distance Vs alpha for the range of alphas
        Find slope of the line for this query
}
```

Figure 2: Pseudo-code for the query perturbation measure

added noise is small compared with the distance between clusters, then a noisy query will still retrieve most, if not all the documents from the original set. As the noise magnitude increases, the query is increasingly likely to retrieve other documents. The rate at which this occurs is used as an approximate measure of the inter-cluster distance.

This approach is similar to the document perturbation approach described in the preceding Section 3. However, there we were attempting to analyze the structure of the retrieved set while here we analyze the structure of the document collection in the vicinity of the query.

This measure is also related to the one described by Yom-Tov *et al* [1] where the authors examine the overlap between the retrieved set due to the query and the retrieved sets corresponding to each sub-query. Sub-queries are obtained from a given query by forcing the weights of certain terms to be zero. In our method the weights are perturbed by a relatively small amount. In either case, a larger degree of overlap between the results of a query and its variants indicates a more stable query whose performance is then predicted as being good.

Let $\boldsymbol{S}$ be a set of N documents retrieved in response to the query $\boldsymbol{q}$. We then perturb every non-zero weight q_k in $\boldsymbol{q}$ by adding noise from a Gaussian of 0-mean and variance $\alpha * v_k$ to generate a new query $\boldsymbol{q}'$. The variance term v_k is calculated from the entire collection of documents. This perturbed query is then issued against the collection retrieving a set $\boldsymbol{S}'$of N documents. The number of elements common to $\boldsymbol{S}$ and $\boldsymbol{S}'$ is an indicator of the query sensitivity. We calculate that statistics across a range of α, i.e., noise magnitudes.

The effect of perturbing the query is reflected in the difference between the retrieved sets for the original and the noisy query. A comparison between these two sets can be performed using a number of measures, the simplest of which are set-overlap statistics like intersection, Jaccard's distance, etc. Since we are interested in document *rankings* produced by queries we also need to account for situations where the retrieved sets are the same but the documents are in differing orders. To measure such differences, we use the edit or Levenshtein distance. The Levenshtein distance between two strings is the number of operations such as insertions, deletions, and substitutions, required to turn one string into the other. We use this distance to measure the sensitivity of the retrieved set to the perturbations of the query.

More precisely, we observe the slope of the graphs that represent the increase in the Levenshtein distance as a function of $\log(\alpha)$ as α increases. We expect this slope to be inversely proportional to the average precision. Again, we use multiple samples (ten) to average possible irregular effects. The pseudo-code for the query perturbation procedure is given in Figure 2.

5. LOCAL INTRINSIC DIMENSIONALITY

In the vector space model, documents are considered points in a high dimensional space with coordinates corresponding to the distinct terms in the collection. However, any given document contains only a small fraction of all the terms. Therefore, while the dimensionality of the entire set of documents is high, the dimensionality of a subspace that a sub-collection of documents occupies can be much smaller. The number of parameters required to represent a set of N points in a D dimensional space is called the *intrinsic dimensionality* and will always be less than $\min(N, D)$.

There is a considerable literature dealing with the calculation of intrinsic dimensionality of a set of points. Fukunaga and Olsen [9] describe a method based on the eigenvalues of local regions in the space occupied by the points. This technique requires the definition of a threshold for *significance* of eigenvalues. Rather than arbitrarily fixing this threshold, we apply Bayesian model

selection using the Laplace criterion by Minka[10] which suggests the optimal number of components to be used for principal component analysis (PCA).

The number of dimensions required to model a given set of documents is an estimate of its "complexity". If we calculate the Laplace criterion for the whole set of documents, it gives us an estimate of the global dimensionality. Inter-document relationships, on the other hand, lead to local groupings. Measuring the number of components needed for each such restricted group gives an estimate of the local dimensionality.

Given the set of retrieved documents, for each point in this set, we identify its closest *K* neighbors within the retrieved set, where *K* ranges from 5 to 20 in steps of 5. The number of components suggested by the Laplace criterion for this set of *K*+1 data points, i.e., the point itself and its *K* neighbors, is the intrinsic dimensionality of that neighborhood.

For a given *K*, we determine the intrinsic dimensionality of the *K* neighborhood of each point in the collection and calculate the average. As we increase K we can observe the rate of change in the intrinsic dimensionality. We use the slope of the increasing intrinsic dimensionality of the retrieved set to predict the search performance. The underlying assumption is that a high dimensional dataset can be decomposed into a lower dimensionality component and noise. If there is a large amount of noise in the data, the number of parameters required to model this essentially random set of points is small. Therefore, the higher the intrinsic dimensionality for a given set of results, the more likely it is that the query is effective.

6. EXPERIMENTS AND RESULTS

In our experiments we use the Lemur toolkit [11] to index and search over TREC disks 4 and 5 after removing standard stopwords. We use the tf-idf weighting where the weight d_{ij} for a term j in document $\mathbf{d}_i$ is given by $\log((N+1)/(n_j+0.5))*t_{ij}$. For each of the 200 TREC topics, 301-450 and 601-650, we use the description field to formulate a query and retrieve the top 100 search results. We consider 100 results for each query with the knowledge that the average number of assessor-judged relevant documents for this set of queries is between 60 and 70. We calculate the average precision for each query from the available relevance judgments. This gives us a "ground truth" ranking of queries according to the search effectiveness as measured by the average precision.

For each query we compute the value of four measures that we described in the previous sessions for the corresponding retrieved set. We rank all 200 queries according to each measure and compare this with the ground truth query ranking. The Kendall τ coefficient between two ranked lists provides a measure of the correlation between them. It estimates the number of pair-wise swaps required to turn one ranking into the other.

Table 1: Correlation between each of the features and the Average Precision

	Clustering Tendency	Document Perturbation	Query Perturbation	Laplace
Kendall τ	0.441	0.521	0.174	0.268

Table 1 provides the Kendall τ correlations between the predicted and actual ranking for each of four features. As can be seen, the document perturbation method provides the best performance with a Kendall τ of 0.521. This compares favorably with [1] which achieves a score of 0.439 on the same database. Moreover, unlike [1], we do not require any learning and assume only a monotonic relationship between our features and the average precision. The other three features also perform well when compared to methods used for the same purpose, such as the use of the standard deviation of IDF of query terms, score of top ranking documents, etc. [1, 2, 3]

6.1 Combining predictive measures

Since each of the four predictive measures captures different properties of the retrieved sets, we expect that combining them should yield a further improved predictive performance. We could construct a problem of learning this ranking over the set of queries by considering labeled examples of pair-wise ordering between queries. In order to avoid the cost of learning, we only consider a simple arithmetic mean of the four measures. However, since each measure has values from different numerical ranges, we normalize them before averaging.

We experimented with three forms of normalization:

1) **The same mean normalization**. Fix one of the measures (e.g., sensitivity to document perturbation) and alter the values of the other three measures so that they all have the same mean. If the measure **Y** is fixed, then all values x of the measure **X** are changed as $x \rightarrow (x \cdot \text{mean}(\mathbf{Y})) / \text{mean}(\mathbf{X})$.

2) **Min-max normalization**. Normalize each measure independently by mapping its value onto the [0, 1] interval. This can be achieved by the mapping: $x \rightarrow (x-\min(\mathbf{X})) / (\max(\mathbf{X})-\min(\mathbf{X}))$, for all values x of the measure **X**.

3) **Inverse tan (arctan) normalization**. Since three of our measures: the document and query perturbation and the change in intrinsic dimensionality represent slopes, we normalize their value by applying the inverse tan (arctan) function and dividing by π/2. This provides a mapping onto the [0, 1] interval. The values for the Cox-Lewis statistic are normalized using the min-max method in 2.

Table 2: Combining four search effectiveness measures

Normalization	Average	Best Achieved
Same mean	0.561	0.561 (Average of all)
Min – Max	0.457	0.550 (Clustering tendency + Document perturbation)
Inverse tan	0.561	0.562 (Clustering tendency + Document perturbation + Query perturbation)

Table 2 shows the performance of the combined estimator for each of the three described normalization approaches. It includes

the Kendall τ correlation between the query ranking based on the arithmetic mean, i.e., the average score of the four normalized measures, and the average precision ranking. Since we can use any subset of four measures for prediction, we also investigated the optimal combination and provided the Kendall τ for the best achieved correlation with the average precision ranking. The normalization does not affect the performance of the individual measures since all three normalization methods are monotonic transformations of the original scores. Thus, the performance of individual normalized measures is the same as shown in Table 1. As expected, a combination of the features is able to achieve better performance than any feature independently.

If we consider the two lists, the actual and the predicted rankings of 200 entries independent, the Kendall τ can be approximated as a normal variable of zero mean and variance 0.0023. This means that our values for the correlation are significant even at the 99.9% confidence level.

6.2 Characterizing queries

One important application of methods for search performance prediction is to flag queries for which the system has not retrieved good search results, even before the results are presented to the user. We thus explore how reliably our methods can be used to distinguish successful from unsuccessful query searches.

For our best performing estimator that uses the average normalized scores of the clustering tendency, document perturbation, and the query perturbation, we assess how well it can detect unsuccessful searches.

We sort the 200 queries in ascending order of the corresponding average precision and consider the queries that fall into the 10%, 20%, 30%, etc., of worst performing queries according to average precision. We rank the queries according to our search effectiveness measure and identify the bottom 10%, 20%, 30%, etc., performing queries according to our measures. We then compute the overlap between the sets of these queries to identify the agreement level. The results provided in Table 3 show that our method can identify unsuccessful searches with a success rate between 55% and 75%.

Table 3: Effectiveness of identifying the poorly performing searches

	20 worst	40 worst	60 worst	80 worst
% correctly identified	55	65	68.33	73.75

7. CONCLUSIONS

In this paper we present methods for estimating search effectiveness by examining properties of the retrieved set and documents in its vicinity. We start with the hypothesis that an effective search will result in a retrieved set that exhibits a structure since relevant documents are likely to be similar and cluster together.

We investigate four measures: the clustering tendency, the sensitivity to the document perturbation and the query perturbation, respectively, and the rate of change in the local intrinsic dimensionality. Three of these measures are focused on examining the original set of retrieved documents while the query perturbation sensitivity examines the structure of the document collection in the vicinity of the query.

Experimental results with TREC disks 4 and 5 and topic sets 301-450 and 601-650 show a significant improvement over the past attempts to predict search effectiveness. We demonstrate that by considering the sensitivity of the retrieved set to document perturbation we can achieve Kendall τ correlation of 0.521 with the average precision ranking of queries. Combining this measure with the clustering tendency, based on the Cox-Lewis statistic, and the query perturbation measure leads to further improvement. We obtain Kendall τ correlation of 0.562 with the average precision ranking. Both of these results are higher than previously reported results for the same document and query collections. Also, our best performing method can correctly detect 65% of the worst 20% searches. This is achieved without a significant computational cost by considering only 100 documents per query.

Our work has some relation to that of [1], particularly with respect to the query perturbation feature. However, an advantage of our method is that it does not require a learning phase. Perhaps more importantly, the Cox-Lewis and the document perturbation measures require computations based only on the retrieved document set.

A pre-retrieval estimate of query effectiveness would be most desirable. However, in view of comparatively low performance of such techniques, we expect that an optimal approach will involve analyzing the retrieved set but with no involvement of the search engine in additional processing. This is particularly important in the case of meta-search engines where analyzing the returned results of each engine is much more feasible than repeated querying. Our measures based on document perturbation and measurement of clustering tendency offer such alternatives. Of course, they require some post-processing of the retrieved sets but the cost is small compared with the cost of new searches.

The measures that we defined and explored are not restricted to estimating search effectiveness. They can be used for comparing the complexity of different document collections and the effects of different document representations on search. In our future work we shall expand the application areas of the existing measures and work on additional ways of predicting the effectiveness of search systems.

8. REFERENCES

[1] E. Yom-Tov, S. Fine, D. Carmel and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In Proceedings of the 28th Annual international ACM SIGIR Conference on Research and Development in Information Retrieval. Salvador, Brazil, 2005

[2] S. Cronen-Townsend, Y. Zhou and B. Croft. Predicting Query Performance. Proceedings of the 25th Annual International ACM SIGIR conference on Research and Development in Information Retrieval. Tampere, Finland, 2002

[3] G. Amati, C. Carpineto and G. Romano. Query difficulty, robustness and selective application of query expansion. In Proceedings of the 25th European Conference on Information Retrieval. Sunderland, Great Britain, 2004

[4] B. He and I. Ounis. Inferring Query Performance Using Pre-retrieval Predictors. In Proceedings of the 11th Symposium on String Processing and Information Retrieval, Padova, Italy, 2004

[5] C. J. van Rijsbergen. Information Retrieval. Butterworths, London, Second Edition, 1979

[6] A. K. Jain and R. C. Dubes. Algorithms for Clustering Data. Prentice-Hall Advanced Reference Series, Year : 1988

[7] T. F. Cox and T. Lewis. A conditional distance ratio method for analyzing spatial patterns. Biometrika 63, 483-491, 1976

[8] A. Tombros and C.J. van Rijsbergen. Query-sensitive similarity measures for Information Retrieval. Invited paper, Knowledge and Information Systems, 2004

[9] K. Fukunaga and D.R. Olsen. An Algorithm for finding intrinsic dimensionality of data. IEEE Transactions on Computers, C-20(2), pp. 176-183, 1971

[10] T. P. Minka. Automatic Choice of Dimensionality for PCA. MIT Media Laboratory Perceptual Computing Section Technical Report No. 514

[11] The Lemur Toolkit for Language Modeling and Information Retrieval, http://www.lemurproject.org/.