Hard Queries can be Addressed with Query Splitting Plus Stepping Stones and Pathways

Xiaoyan Yu
Computer Science Dept.
Virginia Tech
Blacksburg, VA 24061
USA

Fernando Das-Neves
Snoop Consulting
Paraguay 346 Piso 5
Buenos Aires
Argentina

Edward A. Fox
Computer Science Dept.
Virginia Tech
Blacksburg, VA 24061
USA

Abstract

A key finding of the Reliable Information Access Workshop of 2003 was that in collections like those used for TREC 6-8, there are a number of hard queries for which no current search engine can return a high quality set of results. Our Stepping Stones and Pathways (SSP) approach may yield an effective solution to such hard problems, as well as support exploration of collections of content not well known to a person (with broad interest and/or complex information needs). Our initial and promising testing of SSP had users prepare two separate short queries in order to launch processing. However, since beginning with a single information need is a more typical initial situation, we have extended the SSP research by exploring query splitting, especially as might apply to handling hard queries. This paper summarizes our recent results and identifies some of the future work needed.

1 Introduction

Searching, such as of text, is a key service of digital libraries. The quality of search results, however, is highly variable. This situation has been a key concern of the information retrieval community, and also is of interest to the database community.

Though on average results are fairly good, there is room for improvement, and in particular cases, results may be unacceptable. Accordingly, the Robust Retrieval Track of the Text REtrieval Conference (TREC), starting in 2003, has focused on individual query effectiveness rather than average effectiveness [16]. Typically, variability in retrieval effectiveness is caused by: 1) an incorrectly formulated query, 2) a collection that lacks pertinent content, or 3) an information retrieval method/system that is inadequate.

The Reliable Information Access (RIA) Workshop 2003 was initiated to investigate in-depth the reasons for retrieval variability. It approached this by studying the behavior of 7 leading search engines developed by the research community. One of the interesting results is that all of the systems failed on a subset of queries, most of which are considered to be 'hard' due to their multiple-aspect (i.e., touching on several different topics or aspects, each of which should be satisfied) property [8]. This finding suggests that the problem may be due to a combination of causes 1 and 3 listed above.

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

Several studies have tried to predict the variability of effectiveness or to improve the handling of queries for which performance is poor. [11] predicted a query's performance by its clarity score, basically the relative entropy between the query in the relevant document set, versus in the whole document collection set. A SIGIR workshop was held to target the problem, but in summarizing the findings of the meeting, [9] reported that research on query variability prediction is in its infancy. Information retrieval researchers lack a clear understanding of why performance is low for some queries, and cannot reliably predict which queries are 'hard'. Similarly, in the database community, Graupmann targeted this problem; his approach was to add annotations using XML to attach semantic meaning to important terms [12].

Different people have different interpretations of what a document is about. Similarly, people vary in their ability to express their information needs in the form of a query [1]. Based on the finding that the cause of poor performance for some queries might be their multi-aspect property [8], we propose our query-splitting-enabled Stepping Stones and Pathways approach to detect different aspects of a query and then to improve retrieval by identifying connections among these aspects.

We also identify two situations that may exist for some poorly-performing queries:

1. A low-relevance set is retrieved where the results are dominated by a subset of the query aspects.

2. A low-relevance result set is retrieved where it is possible to get high relevance for connected subsets of the query aspects (maybe with modification).

In order to address these two possibilities, we think it necessary to create an alternative interpretation of a user's intention. In this alternative interpretation, a query is identified as a description of two or more separable aspects. By separable aspects we mean that a significantly different set of documents representing each aspect can be retrieved for the query. The query is thus split into multiple sub-queries; without loss of generality we limit the splitting to two [17]. The next step is to retrieve a set of documents that support a valid relationship between the two sub-queries using the Stepping Stones and Pathways (SSP) approach [4]. The result returned by the SSP system is a set of topic sequences (**pathways**). Each step in each pathway is supported by documents connecting a sub-query with an intermediate topic (**stepping stone**), or connecting different intermediate topics, that provide a rationale for the connection between the two original sub-queries, i.e., the **end stones**. The two sub-queries become the endpoints which may be reached through different pathways. Each pathway connects the endpoints by a succession of stepping stones, and thus is an answer to the user information need. The SSP user interface (Figure 1) highlights the end stones, stepping stones, and pathways, and supports as well as exploratory type of search.

Query splitting have been used in the past [7]. Most commonly it converts a natural language query into two parts, so that the new query will be interpreted as of form *X or Y*. However, earlier work on query splitting did not address the problem of coherence from the user's point of view: *Are X and Y related (possibly by other intermediary concepts)?* Ours is the first study we know of that has researched both query splitting and new methods to build and display the connections of the results from query splitting.

The Stepping Stones and Pathways approach was inspired by earlier research on Literature-Based Discovery. The explosion of scientific knowledge in the last half of the 20th century resulted in many researchers being highly specialized. Different researchers may work on related problems without even being aware of each other. Discovering relations that are not explicitly stated but yet are latent in a body of knowledge is the objective of Literature-Based Discovery. Swanson [2] was the first to introduce the idea of discovering such new relations within a bibliographic database. Further work by Swanson with Arrowsmith [3] detected indirect relationships between topics in the Medline database, by finding common keywords between two document sets through an intermediate document set. Our work with SSP has extended the line of research launched by Swanson; we are extending it further through integration with new work on query splitting.

This paper concentrates on the effectiveness of query splitting as a technique for improving retrieval results. The rest of the paper is organized as follows. Our prior work with SSP is summarized in Section 2. Section

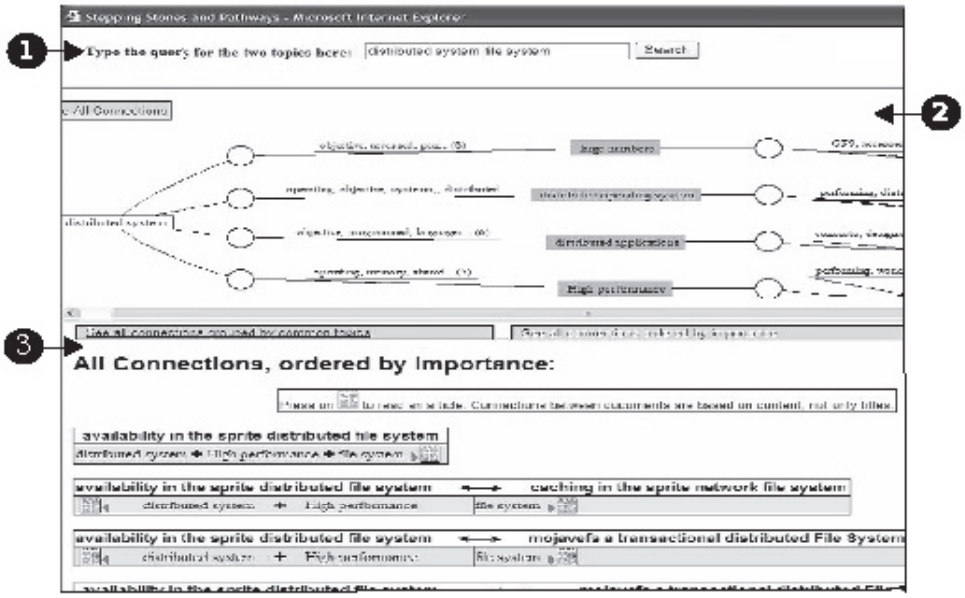

Figure 1: The areas of the Stepping Stones and Pathways user interface.

3 describes query splitting algorithms. Section 4 discusses testing of query splitting methods and the findings from our experiments. Finally, we conclude this paper and list future plans in Section 5.

2 Stepping Stones and Pathways: How it works

In order to describe the context in which we use query splitting to answer hard queries, we briefly discuss how Stepping Stones and Pathways works. In particular, we provide an overview of the user interface and the methods used to create the stepping stones and pathways. Details can be found in [4].

The Stepping Stones and Pathways user interface is divided into three areas (see Figure 1): 1) **The Query Area:** Here the user types a (two-aspect) query describing topics of interest. 2) **The Network Area:** Every time a new query is issued in the query area, this area shows the initial graph connecting the topics in which the user is interested, through a number of intermediate topics. 3) **The Document and Connections Area:** This area displays a list of documents and provides indications of how they support the corresponding connections.

From a user's point of view, a SSP retrieval session starts with the user typing a query. The query is split to create two sub-queries. SSP displays a network, in which the leftmost and rightmost nodes are labeled according to the sub-queries, and intermediate nodes are labeled according to topics connecting the sub-queries. Below the network, SSP also displays a list of documents and explains how each of them supports a connection in the network. The user can click on any node to see all the documents covering that topic, or on any edge to see any connections between the topics at each end of the edge. If the connection between any two topics is too vague, then the user can request SSP to add more intermediate topics (stepping stones) between those topics.

From an implementation point of view, a SSP session works as shown in Table 1.

1. (Off-line) Index the document collection, using the document text and also references, if available.
   Document text is indexed using a tfidf word weighting, after filtering stop words.
2. Split and process the query, trying to match all words in the sub-queries first.
   If that fails, relax the sub-queries by making words optional.
   To calculate the similarity between two documents, use the formula
   $sim(d_1, d_2) = 1 - (1 - P_w(d_1, d_2)(1 - P_{cocit}(d_1, d_2))(1 - P_{ref}(d_1, d_2))$
3. Create the endpoint nodes (end stones) of the graph by:
   3.1. Retrieving two document sets, one from each of the user sub-queries;
   3.2. Creating a document cluster for each document set;
   3.3. Calculate a cluster centroid from the top 10 documents in the cluster;
   3.4. Label the cluster using Suffix-Tree Clustering [6].
4. Create intermediate Stepping Stones and Pathways by:
   4.1. Using the endpoint centroids as queries to find two document sets;
   4.2. Creating an intermediate document set with the documents that appear in both retrieved sets;
   4.3. Finding relevant connections between the documents in the endpoint clusters and
   the documents in the intermediate set;
   4.4. Eliminating all documents in the intermediate set that are not part of a connection;
   4.5. Clustering and labeling documents left in the intermediate set; the clusters become stepping stones.
5. Visualize and display to users the stepping stones and pathways.

Table 1: Overview of SSP working steps

3 Query Splitting

In this section we explain how we decide if a query has multiple aspects. In general, we employ clustering algorithms, heuristic rules, and thresholds set to control the splitting. If there is not enough difference between two clusters, for a given splitting threshold, we call the query a single-aspect one; no splitting is appropriate. On the other hand, when the difference is sufficient, we have identified a multiple-aspect query, and split it into two sub-queries.

3.1 Algorithms for Query Splitting

In order to determine how best to split queries, we have devised and implemented three algorithms.

Relevance-Feedback-Based Clustering (RFC). Borodin [7] also believed a user would search for multiple concepts using a single query. He made good use of relevance feedback information from users, to retrieve more documents for different concepts. We adopt Borodin's method here. q_0 is a user's original query. New queries are generated iteratively as follows.

1. Retrieve the n highest-ranking documents, not previously retrieved, of the current query q_i, for relevance judgment. n is 5 initially.

2. Generate different groups from the documents judged as relevant documents. We put each such pair of documents, d_s and d_t, into different groups if
   $(correlation\, of\, d_s\, and\, d_t) <= \tau\ \times\ [(correlation\, of\, d_s\, and\, q_i) + (correlation\, of\, d_t\, and\, q_i)]/2$.
   Here the cosine correlation is calculated, with τ as the splitting threshold. So there can be zero (if no relevant documents are retrieved), one, or multiple groups.

3. For each group j, get top m words as a new sub-query q^j_{i+1} of the current query q_i by $q^j_{i+1} = q_i + \sum(r)/|R^j_i| - \sum(nr)/|nonR_i|$, where R^j_i are relevant documents of the group j (if none, then omit this expression). $nonR_i$ are at most two non-relevant documents with highest ranking. $r \in R^j_i$ and $nr \in nonR_i$.

4. Repeat the above steps for each newly generated sub-query, if any, but let n be 3 in the first step.

Term-Based Clustering (TC). Based on the observation that the representative terms in a user query might reflect different aspects, we characterize a query using its expanded term list, considering the top retrieved documents. The algorithm basically consists of the following steps.

1. Get the top m words from a query, plus the top $|R|$ retrieved documents for the query, using Rocchio's query expansion algorithm. That is, $q' = \alpha q + \beta \sum(r)/|R|$ where $r \in R$.

2. Represent each word as a list of documents.

3. Calculate the word distances using cosine correlation.

4. Cluster words using the agglomerative hierarchical clustering algorithm, requiring complete linkage.

5. Cut the cluster tree into groups based on the splitting threshold τ, so that each group represents a sub-query.

Document-Based Clustering (DC) Similar to Term-Based Clustering, we also propose an algorithm to cluster a user query based on the diversity of its top retrieved documents. It is like the TC algorithm, but with terms swapped for documents, and vice versa, except that: we use the top k retrieved documents for a query, and in the last step we cut the tree into groups based on the splitting threshold τ, and get the top m terms in the centroid of each group as a sub-query q'. More specifically, $q' = \alpha q + \beta \sum(r)/|R|$, where R represents the documents in a group and $r \in R$.

3.2 Term Scoring Functions and Parameter Values

Term scoring plays an important role in every query splitting algorithm. The different term scoring functions used in the algorithms are: 1) **Term Scoring Functions in First Pass Retrieval:** For all the approaches, we used the Okapi BM25 [13] term scoring function in the first pass retrieval. 2) **Term Scoring Functions in Clustering:** We also used Okapi BM25 [13] for the dividing groups step (i.e., the second step of RFC), to weight query terms and document terms. We tried Okapi BM25 [13] and pivoted tfidf (Ptfidf) [14] for weighting terms in all the clustering parts (i.e., the second step in both TC and DC). 3) **Term Scoring Functions in Query Expansion:** For all the approaches, we employed a Kullback-Leibler Distance (KLD) based method [10] for selecting and weighting expansion terms, with normalization based on dividing by the maximum term weight.

It also is essential to assign values to the parameters properly. Multiple experiments by Carpineto [10] on TREC 7-8 showed that an increasing number of pseudo-relevant documents decreased the retrieval performance nearly monotonically. Further, an increasing number of selected terms in query expansion just slightly increased the retrieval performance. Therefore, we selected $m = 20$ and $|R| = 12$ in the TC method, since that combination yielded good results in [10]. For consistency with the TC algorithm parameter settings, we set $k = 24$, since the ideal case in the DC method is to generate two equal-size clusters. In this case, the number of pseudo-relevant documents in each cluster will be 12, which is the value of $|R|$. For the same reason, we set $m = 10$ in the DC method and the RFC method. Further, $\alpha = 1.0$ and $\beta = 1.5$ are commonly used with the Rocchio algorithm, and so are used in the TC method. For the DC method, we set $\alpha = 0.2$, since we found $\alpha = 1.0$ makes two sub-queries very alike, and to a large extent hides their differences.

$Precision_n$	The minimal value of the rankings, falling within the top n, of each relevant document retrieved by each sub-query.
$Overlap_n$	The maximal value of the rankings, falling within the top n, of each relevant document retrieved by each sub-query.
$Difference_n$	$(Precision_n - Overlap_n)/Precision_n$; This measure is limited to only two sub-queries generated. A more complicated measure should be used to compute the difference among more than two sub-queries.
P_{avg}	The maximum of the P_{avg} values, as used in TREC, for the sub-queries.
$Precision_n$ average, $Overlap_n$ average, $Difference_n$ average or P_{avg} average	The total of the values of $Precision_n$, $Overlap_n$, $Difference_n$, or P_{avg}, divided by the number of detected multiple-aspect queries.

Table 2: A new evaluation strategy.

4 Experiments

In order to evaluate the effectiveness of our approach, three kinds of experiments are required: comparing the three query splitting algorithms in Section 3, evaluating SSP itself, and evaluating the combination of query splitting and SSP. The third experiment is underway. Also, we will not describe here the evaluation of SSP as an effective tool to discover connections among documents and topics, since that is detailed in [5]. Nevertheless, we must recall one of the interesting findings, i.e., that SSP can help users explore many implicit connections between a query pair. In this paper we focus on the first experiment, so as to generate a good split, based on a user's information need, and to provide sub-queries as input for SSP.

4.1 Evaluation Strategy

Since there are no well-known techniques to evaluate the prediction of query difficulty [9], it is hard to apply widely used evaluation strategies when judging the quality of our query splitting algorithms. Also, though it adds complexity to the evaluation problem, we must continue our focus regarding query splitting, wherein we detect poorly-performing queries based on their having the multi-aspect property. Thus, we adopt the requirement of SSP, whose starting point is query splitting. Accordingly, we claim that the quality of a query splitting algorithm depends on three factors: retrieval performance, difference between sub-queries, and overlap of sub-queries.

The most important factor deciding the quality of the results of a sub-query is still the number of relevant documents retrieved. Since the reason to propose a query-splitting algorithm is to improve information retrieval, it is reasonable not to expect a degradation of retrieval performance when using an algorithm. Regarding the second factor, we note that only with enough overlap of sub-queries is it feasible to find the intermediate concepts, i.e., the stepping stones. Regarding the third factor, we observe that a very small difference among sub-queries makes building a bridge among them unnecessary, since the sub-query topics can be directly connected and discussed in a single document to be retrieved by the original query.

We evaluate these three factors based on the retrieval results of the sub-queries. Thus, we make use of the relevance judgment information available in TREC, and evaluate the three query splitting approaches using a $relevance \times rank$ matrix. Each row in the matrix contains all the relevant documents for a specific query; each column corresponds to one query splitting approach; and each cell value is the corresponding relevant document's rank when documents are retrieved by a sub-query generated when the corresponding approach is employed. We define the measures in Table 2 based on the matrices.

	Total number	Total number detected by RFC at the threshold of 12.5	Total number detected by DC and TC at the threshold of 1 using Okapi and Ptfidf	Total number detected by TC at the threshold of 200 using Ptfidf
Queries selected manually	20	14(70%)	20(100%)	16(80%)
Hard queries	17	14(82%)	17(100%)	11(65%)
Union	34	25(74%)	34(100%)	24(71%)
Overlap	3	3(100%)	3(100%)	3(100%)

Table 3: Query split results using different algorithms

4.2 Experimental Setup

Collections and Queries. Our test collection is the one used in the Robust Track of TREC 2004 [15]. Two sets of queries were chosen from the TREC queries. The first set consists of 17 queries selected due to the multiple-aspect property pointed out in the analysis in [8, 15]. The other set consists of 20 queries we selected as likely to have multiple aspects, based on reading the title, description, and narrative. There are three queries that are common to both sets; thus we have 34 unique queries.

Upper Bound Experiment. Since we do not know in advance what should be the proper splitting threshold in each algorithm, we ran an upper-bound experiment first using trial-and-error to find the threshold value under which the corresponding algorithm performs the best. Then we compared the algorithms using their optimal settings. We evaluated each algorithm's performance based on the evaluation strategy in Table 2. More specifically, the metrics are $num_{detected}$ (the number of detected multiple-aspect queries), p_{avg}, sum_n ($precision_n$ + $overlap_n$ + $difference_n$), where $n = 20$, 30, 50, and 100. We consider each metric to be of the same importance, so we normalize each one by its total value. The higher the value of each metric, the better the algorithm performs.

4.3 Findings

Splitting Thresholds. By the upper bound experiment, we identify the optimal settings for each algorithm: 1) RFC: the splitting theshold $\tau = 12.5$; 2) TC when using Okapi as the term scoring function in clustering: $\tau = 1$; 3) TC when using Ptfidf as the term scoring function in clustering: $\tau = 200$; 4) DC when using Okapi as the term scoring function in clustering: $\tau = 1$; 5) TC when using Ptfidf as the term scoring function in clustering: $\tau = 1$. More details are in [17].

Query Splitting Results. We summarized the number of queries split by each algorithm in their optimal settings in Table 3. At least 70% of the queries that we selected by manually judging the multi-aspect property are split by all the algorithms. At least 65% of the hard queries identified with multi-aspect property by [8] are split as well. All the algorithms split all the common three queries. Further, we note that all the algorithms split at least 24 of the 34 queries (71%).

Comparison of Best Cases of All the Algorithms. In general, the results follow the pattern $P_{RFC} > P_{DC} > P_{TC}$, where P stands for performance, as can be seen in Figure 2.

We also measured the performance of retrieval of the original query without a splitting process, and the refined query with a query expansion (QE) process, using the p_{avg} average, since other metrics are not applicable.

original	QE	TC (w=Okapi,τ=1)	TC (w=Ptfidf,τ=200)	DC (w=Okapi,τ=1)	DC (w=Ptfidf,τ=1)	RFC (τ=12.5)
0.157597	0.171934	0.155587	0.156439	0.179454	0.175758	0.26489

Table 4: Comparison of all the best cases with the retrieval performance by original queries without query splitting and the performance by refined queries with only query expansion (QE) in terms of p_{avg}.

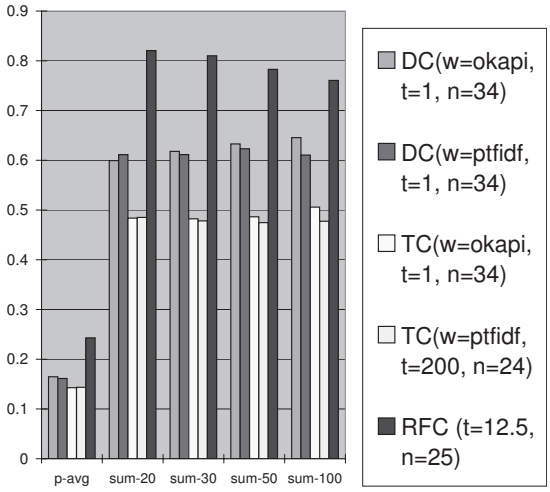

Figure 2: Comparison of all the best cases on the metrics of p_{avg}, sum_{20}, sum_{30}, sum_{50}, and sum_{100}.

The refined queries are produced using the method in [10], which is also the method to select top terms in all the query splitting algorithms. The results are shown in Table 4. Considering only the p_{avg} average, the performance of the TC algorithm is very close to the original query retrieval performance, while the DC algorithm using Okapi and Ptfidf are 13.9% and 11.5% better than the original query performance. The RFC algorithm is even better, 68.1% over the original query retrieval performance. The refined queries perform 9.1% better than the original ones, on average, and also are close to the DC algorithm.

When using different term weighting mechanisms, the performance values for the same algorithm are very close to each other, except concerning the metric sum_{100}. For the DC algorithm, the result when using Okapi is 5.7% better than when using Ptfidf; for the TC algorithm, the one using Okapi is 6% than when using Ptfidf. However, there are more interesting results when we consider the $precision_n$, the $overlap_n$, and the $difference_n$, respectively. Table 5 shows the results and reveals a pattern: the TC algorithm gets a rather low value on the overlap metric, which leads to a relative low result of sum_n.

	P_{20}	O_{20}	D_{20}	P_{30}	O_{30}	D_{30}	P_{50}	O_{50}	D_{50}	P_{100}	O_{100}	D_{100}
DC(w:Okapi,τ:1)	0.197	0.249	0.153	0.203	0.255	0.160	0.214	0.266	0.153	0.216	0.278	0.152
DC(w:Ptfidf,τ:1)	0.193	0.247	0.172	0.202	0.250	0.159	0.201	0.268	0.154	0.201	0.273	0.137
TC(w:Okapi,τ:1)	0.171	0.068	0.245	0.174	0.051	0.257	0.178	0.041	0.268	0.184	0.037	0.285
TC(w:Ptfidf,τ:200)	0.164	0.069	0.253	0.160	0.057	0.261	0.160	0.045	0.270	0.163	0.034	0.280
RFC(τ:12.5)	0.275	0.368	0.177	0.261	0.386	0.163	0.247	0.380	0.155	0.236	0.379	0.146

Table 5: Comparison of all the best cases on the P_i($precison_i$), O_i($overlap_i$), and D_i($differece_i$) metrics, where $i = 20,\ 30,\ 50,\ 100$.

4.4 Discussion

The term-based clustering and document-based clustering algorithms perform better, if not the best, when all the queries are split. Even considering the relevance-feedback-based clustering, there is a trend that the performance is better when more queries are split. More importantly, as can be seen in Table 2, the performance (measured by p_{avg}) of each algorithm in its optimal settings is not worse, and sometimes is better, than when there is no splitting. Also, all of these algorithms, except for TC, perform even better than does the refined query resulting from query expansion. Hence the majority of the query samples we selected are, indeed, multi-aspect queries, since the splitting process did not hurt performance. We expect that integrating query splitting with SSP will yield an even better result since SSP can find more relevant documents by means of discovering the connections within a multi-aspect query.

Relevance-feedback-based clustering is a special variation of the document-based clustering, since its basic idea is to cluster documents and use a cluster centroid as a sub-query. However, it clusters already-known relevant documents (from relevance judgments) instead of top retrieved documents. Hence much less noise should be included when representing the aspects of an original query. However, in practice, this kind of relevance information only can be obtained implicitly or explicitly from users. How to collect such information accurately, but not intrusively, is still an open question.

In our experiments document-based clustering performed in general better than term-based clustering. As we pointed out in Section 4.3, the utility of term-based clustering for splitting was poor due to its rather low overlap metric value. Term-based clustering divides the term candidate representatives for a query so that there are no overlap terms in the sub-queries of the query, hence decreasing the probability of the overlap of top retrieved results. On the other hand, the document-based approach clusters the document candidate representatives for the query and generates sub-queries containing common terms, from a document cluster. Since the query splitting results are to be fed into SSP, which finds connections between the query parts, we expect that different enough sub-queries will yield a bigger search space for intermediate concepts as bridges connecting the sub-queries. Our future experiments on the combination of SSP and document-based clustering and term-based clustering, respectively, should yield further insight.

The term scoring function used had no significant effect on the algorithm performance, though Okapi results generally were slightly better than Ptfidf. Consequently, in future work, we will use Okapi as the term scoring function for clustering (when testing performance on the combination of SSP and query splitting).

5 Conclusions and Future Work

We have studied an approach to handle poorly-performing queries where a possible reason for the poor results is their multi-aspect property. We have shown the feasibility of splitting this type of queries without decreasing the retrieval performance.

We plan further experiments to test how much retrieval improvement will result from using SSP, taking the split results as input. The experiments will consist of automatic runs and user studies. We will run SSP on the TREC collection, already used in the query-splitting experiment, and will evaluate the results using p_{avg} and other reasonable measures. It is also important to get feedback from real users with respect to their subjective impression of the query splitting results. Accordingly, we plan a user study on the split results, and also one on SSP, with those results as input.

6 Acknowlegments

Our work was funded in part by NSF grant IIS-0307867.

References

[1] Cleverden, C.W. (1991). The Significance of the Cranfield Tests on Index Languages. In Proceedings of ACM SIGIR91, pp. 3-12, ACM Press, 1991.

[2] Swanson, R. (1986). Fish oil, Raynauds syndrome, and undiscovered public knowledge. Perspectives in Biology and Medicine, 30(1): 7-18.

[3] Swanson, R., Smalheiser, N, Bookstein, A. (2001). Information Discovery from Complementary Literatures: Categorizing Viruses as Potential Weapons. Journal of the American Society for Information Science and Technology, 52(10): 797-812.

[4] Das Neves, F. (2004). Stepping Stones and Pathways: Improving Retrieval by Chains of Relationships between Documents. Ph.D. dissertation, http://scholar.lib.vt.edu/theses/available/etd-11012004-003013/restricted/dissertation.PDF.

[5] Das Neves, F., Fox, E.A., and Yu, X. (2005). Connecting topics in document collections with Stepping Stones and Pathways. In Proceedings of the 2005 ACM CIKM, Bremen, Germany, 31 October - 5 November.

[6] Zamir, O., Etzioni, O. (1998). Web document clustering: a Feasibility Demonstration. Proceedings of SIGIR98, pp. 45-54. ACM Press.

[7] Borodin, A., Kerr. L., Lews, F. (1968). Query Splitting in Relevance Feedback Systems. Scientific Report No. ISR-14, Dept. of Computer Science, Cornell University, Ithaca, NY.

[8] Buckley, C. (2004). Why current IR Engines Fail. In Proceedings of SIGIR2004. ACM Press.

[9] Carmel, D., Yom-Tov, E., and Soboroff, I. (2005). Predicting Query Difficulty: Methods and Applications. In SIGIR 2005 workshop.

[10] Carpineto, C., et al. (2001). An information-theoretic approach to automatic query expansion. ACM Transactions on Information Systems (TOIS), 19(1): 1-27.

[11] Cronen-Townsend, S., Zhou, Y., and Croft, W.B. (2002). Predicting Query Performance. In SIGIR 2002. Tampere, Finland: ACM.

[12] Graupmann, J., Schenkel, R., and Weikum, G. (2005). The SphereSearch Engine for Unified Ranked Retrieval of Heterogeneous XML and Web Documents. In VLDB.

[13] Robertson, S.E., Walker, S., and Beaulieu, M. (1999). Okapi at TREC-7: automatic ad hoc, filtering, VLC and interactive. In Proceedings of the 7th Conference on Text Retrieval. Gaithersburg, MD.

[14] Singhal, A., Buckley, C., and Mitra, M. (1996). Pivoted document length normalization. In Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval. pp. 21-29.

[15] Voorhees, E.M. (2005). The TREC Robust Retrieval Track. ACM SIGIR, June, p. 39.

[16] Voorhees, E.M. (2003). Overview of the TREC 2003 robust retrieval track. In Proceedings of TREC 2003. Gaithersburg, MD.

[17] Yu, X., Das Neves, F., and Fox, E.A. (2005). Query Splitting. Virginia Tech Department of Computer Science Technical Report, November.