Query Expansion For Handling Exploratory And Ambiguous Keyword Queries

by

Sivaramakrishnan Natarajan

A Thesis Presented in Partial Fulfillment
Of the Requirements for the Degree
Master of Science

Approved April 2011 by the
Graduate Supervisory Committee:

Yi Chen, Chair
Selcuk Candan
Arunabha Sen

ARIZONA STATE UNIVERSITY

May 2011

ABSTRACT

Query Expansion is a functionality of search engines that suggest a set of related queries for a user issued keyword query. In case of exploratory or ambiguous keyword queries, the main goal of the user would be to identify and select a specific category of query results among different categorical options, in order to narrow down the search and reach the desired result. Typical corpus-driven keyword query expansion approaches return popular words in the results as expanded queries. These empirical methods fail to cover all semantics of categories present in the query results. More importantly these methods do not consider the semantic relationship between the keywords featured in an expanded query. Contrary to a normal keyword search setting, these factors are non-trivial in an exploratory and ambiguous query setting where the user's precise discernment of different categories present in the query results is more important for making subsequent search decisions.

In this thesis, I propose a new framework for keyword query expansion: generating a set of queries that correspond to the categorization of original query results, which is referred as *Categorizing query expansion*. Two approaches of algorithms are proposed, one that performs clustering as pre-processing step and then generates categorizing expanded queries based on the clusters. The other category of algorithms handle the case of generating quality expanded queries in the presence of imperfect clusters.

DEDICATION

To my thatha, amma, appa, my brother and bhuvan.

ACKNOWLEDGMENTS

I would like to acknowledge the enthusiastic supervision of Dr. Yi Chen for her guidance, encouragement and for providing the opportunity to work on several challenging research project including this thesis work. I would also like to extend my sincere gratitude to her for guiding me through the process and providing me with some great opportunities of being a significant part of many of her research projects by believing in my potential. It is difficult to truly express by words, how thankful I am for the motivation I received from her and the transformation it had in my character.

I also thank my committee members, Dr. Selcuk Candan and Dr. Arun Sen, for their time and effort to point out potential improvements and help me fulfill the degree requirements. I would like to thank my lab mate and friend Ziyang Liu for his inspirational presence, continuous support and guidance through the project.

This document would be incomplete if I forget to mention how grateful I am to my friends at Arizona State University for making my stay in Tempe, Arizona particularly enjoyable - my ASU friends, Ganesh Jayachandran, Rohit Raghunathan, Kumaraguru Paramasivam, Rajagopalan Ranganathan, Dananjayan Thirumalai, my lab mate Yichuan Cai, my friend since childhood, Vijayendran Gurumoorthy and my new roommate Madhan Kumar Kuppusamy.

Finally, I am indebted to my girlfriend Bhuvana Kannan and my parents for their encouragement, understanding and belief in my potential. This work would have not been possible without their patience and warmth.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

1.1 MOTIVATION

Web search engines typically make query suggestion based on similar and popular queries in the query log [Chirita et al. 2007; Bar-Yossef and Gurevich 2008]. To handle a bootstrap situation where the query log is not available, there are works on query result summarization [Xu and Croft 1996; Carpineto et al. 2001; Cao et al. 2008; Tao and Yu 2009; Koutrika et al. 2009; Sarkas et al. 2009], where popular words in the results are identified and suggested to the user for query refinement. The popularity of words are typically measured by factors such as term frequency, inverse document frequency, ranking of the results in which they appear, etc.

Result summarization based approaches using popular words cannot effectively capture multiple classifications of results. The problem becomes especially severe when the popular words are obtained from top k results, which is typically the case for efficiency reasons. In such cases, one type of results may have higher ranks and may suppress other result types to be reflected in the expanded queries. For instance, when searching "Tennis" on Bing, top 50 results are about the "game Tennis", but there are also other interpretations of "Tennis" such as a "place in Egypt" or a "music album" of the same name. These results never appear as part of the top k results. Result summarization based approaches thus cannot effectively handle ambiguous or exploratory queries [Broder 2002] where the users don't have a specific search target, but would like to navigate the space of possibly

relevant answers and iteratively find the most relevant ones by refining the results.

To handle ambiguous and exploratory queries, ideally query expansion should provide a categorization of different interpretations of the original query and thus guide the user to refine the query in order to get more results of the desired type. One typical way to approach this problem is by clustering the result set online and generating cluster labels by cluster summarization based methods [Carmel et al. 2009, Cutting et al. 1992]. Typically these approaches give labels to each cluster by adopting differential or internal cluster labeling techniques. This includes techniques such as mutual information based term selection methods [Geraci et al. 2007, Manning et al. 2008] which select terms that commonly occur in results of the same cluster in contrast with other clusters and frequency based term selection methods [Carmel et al. 2009, Cutting et al. 1992] which select highly frequent terms in the same cluster in contrast with other clusters.

However these works overlook several problems associated with query expansion. One problem is that, they may find keywords that occur frequently in fewer documents in the cluster and thus may not cover many results in the cluster. Another problem with such approaches is that, they don't consider the semantic relationship between the keywords in an expanded query. An expanded query generated by cluster summarization based approaches may be composed of keywords from disparate topics which may have low co-occurrence together. For example: A suggested expanded query such as "*Canon Cameras, Printer: Type: InkJet Color*" for a user issued query "*Canon*

Products" may be of little help to the user, especially when the user is trying to explore the results and narrow down the search to find a desired result. This problem becomes a more common occurrence when the clustering quality is poor and when results corresponding to multiple semantics is grouped into same cluster. Since cluster summarization based approaches completely ignore clustering quality into consideration, they generate labels that may reflect the cluster composition but yet have semantically poor quality. While using sophisticated clustering methodology to improve cluster quality would be a direction to follow, it nevertheless guarantees the consistent generation of quality expanded queries.

Therefore, this problem illustrates a unique challenge in generating queries for clustered results; the interaction of keywords must be considered. Moreover, a potentially large number of results, and a large number of distinct keywords in the results add further challenges to the problem. Exhaustively searching for the optimal query for each cluster will be prohibitively expensive in practice. The problem of generating optimal set of expanded queries given the ground truth of query results is shown to be NP-hard and also APX-hard (i.e., it does not have a constant approximation).

In this work, I propose two categories of algorithms to generate expanded queries that are comprehensive and diverse in covering different classifications in the results. The first category of algorithms, as an initial step clusters the results and then generates expanded queries for each cluster that can retrieve maximal results from the same cluster and minimal results from other clusters. If we consider the cluster of results as the ground truth, the

goal is then to generate a query whose results achieve both a high precision and a high recall. This category of algorithms proposed is named as "Query Expansion using clusters (QEC)".

If the clustering quality is good enough, the expanded queries should provide a good categorization of the original query results. However, this may not be the case since the clustering algorithms often generate imperfect clusters, e.g., the clustering quality of k-means method can be sensitive to the initial set of "means". Unless the initial means are chosen carefully to represent results from different classifications, the clusters might not aid towards generating diverse expanded queries. For example, suppose k = 2 and if 99% of results are about Apple company and only 1% is about apple fruit, unless one of the initial means is the apple fruit result, k-means will never put the apple fruit result into a single cluster. The second category of algorithms proposed has two approaches, one which extracts comprehensive and diverse expanded queries even in the presence of noisy or imperfect clusters and another approach which directly generates expanded queries given the original query results. This category of algorithms is named as "Categorizing Query Expansion (CQE)".

1.2 CONTRIBUTION

The contributions of this work include:

- A new problem for query expansion is proposed which aims at providing a categorization of query results dynamically based on query results. This especially has its application on handling exploratory and ambiguous queries.

- Different from the empirical methods of cluster/result summarization, this work proposes a new philosophy of considering interactions between keywords for generating meaningful categories of expanded queries.
- Two kinds of approaches are proposed; generating expanded queries based on result clusters and in the presence of noisy clusters. Two algorithms are proposed for each approach, which generate meaningful expanded queries efficiently.
- Evaluation measures to quantify the quality of a set of expanded queries are proposed.
- The quality and efficiency of the proposed approaches have been verified in the experiments using real datasets. Comparison experiments with some of the main body of existing works and a large scale user study has also been performed.

CHAPTER 2 PROBLEM DEFINITION

As mentioned earlier, the goal of this work is to generate a set of expanded queries that provides a classification of possible interpretations of the original user query. The input includes a user query and a set of query results where the results are optionally ranked.

2.1 QUERY EXPANSION USING CLUSTERS (QEC)

To generate a set of expanded queries corresponding to a classification of the original query, a natural way is to first cluster the query results using an existing clustering method. Then one expanded query is generated for each

cluster, which maximally retrieves the results in the cluster, and minimally retrieves the results not in the cluster. In this way, considering the cluster as the ground truth, the quality of an expanded query can be measured using precision, recall and F-measure. Precision measures the correctness of the retrieved results, recall measures the completeness of the results, and F-measure is the harmonic mean of them.

Let $C1, \ldots, Ck$ denote the set of result clusters, qi denote the query generated for cluster Ci $(1 \leq i \leq k)$, $R(qi)$ denote the set of results of qi. The precision, recall and F measure of qi are computed as,

$$precision(qi) = \frac{R(qi) \cap Ci}{R(qi)}, \qquad recall(qi) = \frac{R(qi) \cap Ci}{Ci} \tag{1}$$

$$precision(Q \cup Ki) = \frac{R(Q \cup Ki) \cap Ci}{R(Q \cup Ki)}, \qquad recall(Q \cup Ki) = \frac{R(Q \cup Ki) \cap Ci}{Ci}$$

$$F\ measure = \frac{2 \times precision(qi) \times recall(qi)}{precision(qi) + recall(qi)} \tag{2}$$

To handle the general case where the results are ranked, weighted version of precision and recall is used. Let $S(.)$ denote the total ranking score of a set of results, then

$$precision(qi) = \frac{S(R(qi) \cap Ci)}{S(R(qi))}, \quad recall(qi) = \frac{S(R(qi) \cap Ci)}{S(Ci)} \tag{3}$$

The F measure defined above serves as an evaluation measure for individual expanded queries. For evaluating the overall quality of the set of expanded queries generated, the harmonic mean of the F-measures is used, while other aggregation measures (e.g., algebraic mean) can also be used.

$$score(q1, q2, \dots, qk) = \frac{n}{\frac{1}{F\ measure(q1)} + \dots + \frac{1}{F\ measure(qk)}} \quad (4)$$

To summarize, the problem of generating expanded queries based on clustered results with the assumption that clusters would be reliable is defined as follows:

Definition 2.1: *Given a set of clusters of query results,* $C1, \dots, Ck$*, the Query Expansion with Clusters problem (QEC) is to find a set of queries, one for each cluster, such that their score (Eq. 4) is maximized.*

Note that the QEC problem is APX-Hard. [Appendix C]

2.2 CATEGORIZING QUERY EXPANSION (CQE)

There is an inherent disadvantage of the cluster based approach of section 3.1: it considers the clustering algorithm as a black box, thus the quality of that method will be much dependent on the quality of the clustering. The following example shows how undesirable clusters may prevent us from getting desirable results.

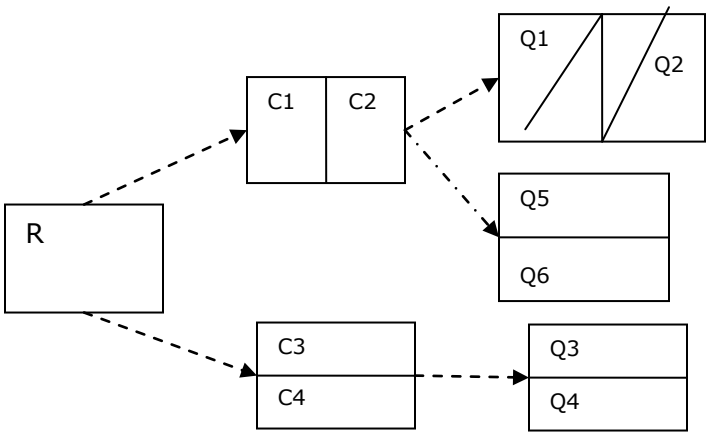


Figure 1. Effect Of Undesirable Clusters.

Consider the above example figure, where there could be 2 ways to cluster the result set R using classical K means, based on different initial means. The clusters C3 and C4 allow generating desirable queries Q3 and Q4 achieving maximum score as per Eq. 4. With clusters C1 and C2, the best queries that could be generated are Q1 and Q2, which have bad recall although the precision is good. Note that with clusters C1 and C2, it is not possible to generate queries Q3 and Q4 as they have both bad precision and bad recall in this case according to the goal function defined in Eq. 4.

From the above example, it could be seen that the clusters which are considered as ground truth in case of Eq. 4, can no longer serve as ground truth. In this case, an optimal set of expanded queries should cover maximal results in the result set and should have minimal overlap with one another. For the above example, the optimal queries could be Q5 and Q6 that cover all the results in the result set contrary to Q1 and Q2 that misses out some results and also have no overlap between them.

From the above intuition, the below definitions of coverage and overlap are derived and finally the score of the set of expanded queries that should be maximized for handling imperfect clusters is defined.

$$coverage(q1, q2, \dots, qn) = \frac{\bigcup_{1 \le i \le n} R(qi)}{\boldsymbol{R}} \quad (5)$$

$$overlap(qi, qj) = \frac{R(qi) \cap R(qj)}{R(qi) \cup R(qj)} \quad (6)$$

$$overlap(q1, q2, \dots, qn) = AVG_{1 \le i \le j \le n} overlap(qi, qj) \quad (7)$$

Considering large coverage and small overlap as desirable, the overall score of set of expanded queries is defined as:

$$score(q1, q2, \dots, qk) = \frac{2 \times coverage(q1,\dots,qk) \times \left(1 - overlap(q1,\dots,qk)\right)}{coverage(q1,\dots,qk) + \left(1 - overlap(q1,\dots,qk)\right)} \quad (8)$$

Definition 2.2: *Given a set of results R, retrieved by a user query, the Categorizing Query Expansion problem (CQE) is to find a set of queries, such that their score (Eq. 8) is maximized.*

Note that the problem of CQE is challenging than QEC problem, since in the QEC problem, each expanded queries can be generated independently. This is because maximizing the overall score (Eq. 4) is equivalent as maximizing the F-measure of each query. On the other hand, an algorithm for the CQE problem needs to determine the number of expanded queries, as well as consider the interactions of different queries during the generation of expanded queries.

CHAPTER 3 ALGORITHMS FOR QEC PROBLEM

As mentioned before, the algorithms proposed for QEC problem is based on clustered query results. Incremental K means [Manning et al. 2008] is used as the clustering algorithm, which dynamically determines K based on best cluster quality evaluated by a goal function. Specifically, the algorithm increments the value of K in each iteration and terminates when the rate of change of goal function flattens. Here, the goal function is a linear function of (1) benefit, determined by the sum of intra cluster similarity of each result to its cluster centroid and (2) cost, determined by weighted function of the number of clusters. The reason for choosing K means is mainly due to its efficient linear time complexity. Since the clustering needs to be done online based on user's query, the clustering algorithm needs to be simple, faster and should not affect the efficiency of overall time taken to generate expanded queries.

3.1 ITERATIVE SINGLE KEYWORD REFINEMENT (ISKR)

The first algorithm is named as Iterative Single-Keyword Refinement (ISKR). Given the user query and a cluster of results, the ISKR algorithm iteratively refines the input query until it cannot further refine the query to improve the F-measure of the query result (considering the cluster as the ground truth). Then, it outputs the refined query as the expanded query for the cluster. Specifically, the algorithm quantifies a value for each keyword appearing in the results, and refines the query by choosing the keyword with the highest value found in each iteration.

There are several challenges that need to be addressed to make this approach work: (1) How to quantify and compute the value of each keyword, (2) Keywords interact with each other, as one keyword is added to the expanded queries, the values of other keywords may change based on the results retrieved by the added keyword. How to identify the keywords that are affected and update the values of these keywords? (3) Starting with a user query, the algorithm iteratively adds keywords to it, while doing so, it could be possible that the F-measure might improve by removing some already added keywords, how to handle such cases? (4) Since there can be potentially large number of results, there can be a large number of distinct keywords, how can we ensure efficiency in picking the right keywords and maintaining the values of each keywords?

Value of a keyword: As each keyword is added to the expanded query the F-measure of the expanded query increases or decreases. Ideally, the effect of adding a keyword can be measured by the delta F-measure of the keyword. But the disadvantage of delta F-measure is that it is hard to maintain as each time when a keyword is added, the delta F-measure of every keyword needs to be updated. Note that, as each keyword is added, two things happen: Results are eliminated from other clusters which is a positive effect, and results are also eliminated from the same cluster which is a negative effect. Therefore the value of a keyword can be measured by the number of results eliminated from inside and outside the cluster. It is to be noted that, this measure is relatively easier to maintain as we only have to update the values of a subset of keywords.

Example 4.1.1: Let "D" denote the set of results eliminated by adding keyword k to the current expanded query q. If a keyword k' is present in all results in D, then it cannot eliminate any results in D. Therefore, the delta results of k' with respect to q are the same as delta results of k' with respect to $q \cup \{k'\}$, since the number of results eliminated by k' never gets affected when k gets added to q. From this example, we can observe that in each iteration as a new keyword is added to the expanded query; only a subset of keywords that can eliminate results retrieved by the current expanded query has to be updated with a new value.

With these observations, we measure the value of a keyword by benefit and cost. Benefit (k, q) is the total score of the results eliminated in other clusters U, and cost (k, q) is the total score of results eliminated from the same cluster C.

$$benefit(k,q) = S(R(q) \cap U \cap E(k)) \quad (9)$$

$$cost(k,q) = S(R(q) \cap C \cap E(k)) \quad (10)$$

The value of a keyword k with respect to q is measured by the ratio of benefit and cost.

$$value(k,q) = benefit(k,q) - cost(k,q) \quad (11)$$

Example 4.1.2: The following example illustrates the steps involved in the ISKR algorithm, psuedocode for all algorithms are provided in appendix B.

Suppose, the user's query is "Apple", the example below shows the process involved in ISKR when it tries to generate expanded query for one of the clusters C having results $R1, \ldots, R8$. The results in other clusters U are

$R1', \ldots, R10'$. The table below shows the various keywords and the results they eliminate from the same cluster C and from other clusters U. The algorithm iteratively adds each keyword with the goal of maximizing the number of results eliminated from other clusters and minimizing the results eliminated from the same cluster by choosing the keywords by their benefit cost ratio.

Table 1

Values of Keywords Computed By ISKR:

Ki	$E(Ki) \cap U$	$E(Ki) \cap C$	Benefit	Cost	Value
Job	$R1', \ldots, R8'$	$R1, \ldots, R6$	8	6	4
Store	$R1', \ldots, R4', R9'$	$R1, \ldots, R4$	5	4	1
Location	$R5', \ldots, R8', R10'$	$R2, \ldots, R5$	5	4	1
Fruit	$R2', \ldots, R4'$	$R1, \ldots, R3$	3	3	0

Since keyword "Job" has the highest value, it is first added. As a result, the values of other keywords get affected, for example: the keyword "store" no longer eliminates the results $R1, \ldots, R4$ in C and $R1', \ldots, R4'$ in U. The updated table looks like below:

Table 2

Updated Values Of Keywords:

Ki	Benefit	Cost	Value
Job	6	8	-2
Store	1	0	1
Location	1	0	1

Fruit	0	0	0

After multiple iterations, the keywords "store" and "location" also gets added to the expanded query and the updated table looks like below:

Table 3

Updated Values Of Keywords:

Ki	Benefit	Cost	Value
Job	1	0	1
Store	0	1	-1
Location	0	1	-1
Fruit	0	0	0

Necessity for keyword removal:

It is possible that sometimes, removing an already added keyword will be beneficial, for instance, in the above example, removing the keyword "Job" is now beneficial, as the current expanded query {"apple", "job", "store", "location"} retrieves two results from C. Removing "job" would retrieve one more result in C, without involving any cost. Therefore the keyword "job" is removed leading to the expanded query of {"apple, "store", "location"}.

The benefit and cost of removing a keyword is computed based on the number of results that will be added back to D (k), when a keyword is removed, as given below:

$$benefit(k, q) = S(D(k) \cap C) \quad (12)$$

$$cost(k, q) = S(D(k) \cap U) \quad (13)$$

Note that, in contrast to the addition case, removal of keyword increases the number of results in both U and C. Thus the removal of a keyword increases recall (measured by benefit) and decreases precision (measured by cost). The value of the keyword is computed similar to the addition case by the benefit cost ratio. The ISKR algorithm stops when the best value keyword has a value of zero or below.

In the implementation, in order to efficiently retrieve the best value and to update the values of other keywords often, the keywords and their values are stored in balanced binary search tree.

3.2 PARTIAL ELIMINATION BASED CONVERGENCE(PEBC)

ISKR algorithm iteratively adds/removes keywords because of which the values of many other keywords need to be maintained which is cumbersome. Intuitively, all we need to find is an expanded query that maximizes the F-measure of precision and recall. The second algorithm tries to find directly the set of keywords that has the best F-measure. However, since the space of all possible queries is exponential to the data size, finding such a query is challenging. In this algorithm, I propose to first select a set of sample data points in the search space, and choose the most promising set of points among them and continue to search for more data points within this range with the goal of further improving the f-measure.

Specifically, given a set of queries and their f-measure, set of queries with highest average f-measure is chosen and then the algorithm proceeds by

finding a better query within the f-measure range of these chosen set with the assumption that optimal queries may exist in this range. This method is closely related to interpolation in numerical analysis, which helps in constructing new data points within the range of a set of discrete known points. Now two questions need to be answered, what kind of sample points should we use to converge to optimal solution and how do we obtain such points?

Type of sample queries: To answer the first question, a set of sample queries can be used, each of which maximizes the number of results to be retrieved in C, given a percentage of results in U to be eliminated. This is in the spirit of maximizing the recall given a fixed precision. If result ranking is not present, the approach aims at eliminating x% of U's results; otherwise, it aims at eliminating a set of U's results, such that their total ranking score is x% of the total ranking score of all the results in U. In the following, I use "x% of the results in U" to refer in general to both cases.

Example 4.2.1: Consider that the algorithm generates five queries, q1 to q5, to eliminate 0%, 25%, 50%, 75% and 100% of the results in U respectively, and maximize the number of results in C to be retrieved. Suppose the F-measures of these queries are: 0.5, 0.6, 0.8, 0.4 and 0.1 respectively. The algorithm takes the two adjacent queries whose average F-measure is the highest, which are q3 and q4 and zooms in this interval between 50% and 75% further dividing them to several intervals and repeating the process.

Generating sample queries: The key challenge of PEBC algorithm is how can we eliminate roughly x% of the results in U and maximize the number of

retrieved results in C. This problem is referred to as *Partial Elimination*. This problem bears some similarity to weighted partial set cover problem, which aims at finding a set of subsets with the lowest total weight to cover at least x% of the elements in the universal set. However, in contrast to the partial weighted set cover problem which requires to cover at least x% of the elements, the goal is to eliminate as close to x% of the elements as possible. Some of the methods studied for generating sample queries are discussed below.

3.2.1. Generating queries based on benefit-cost.

One intuitive method is to apply the greedy algorithm commonly used in weighted set cover for keyword selection: each time, select the keyword with the largest value based on benefit-cost, until approximately x% of the results in U are eliminated. Benefit and cost are defined in the same way as in ISKR: benefit is the total weight of the un-eliminated results in U that a keyword can eliminate, and cost is the total weight of the un-eliminated results in C that a keyword can eliminate.

However this method has an inherent problem. The keyword values based on benefit and cost do not change with varying x; the keywords are always selected in the same order. Specifically, let the list of keywords selected when x = 100 be K = k1,…,kp. Now we want to select keywords to generate a query for each point in a range of possible values of x. No matter which point it is, the set of keywords selected will be a prefix of K. This "fixed-order" selection of keywords makes it very difficult to control the percentage of results being eliminated.

Example4.2.1.1: Consider a total of 10 results in U,$R1, \dots, R10$, and 4 keywords k1="job", k2="store", k3="location", k4="fruit". Suppose the set of results eliminated in U by each keyword (benefit) and the number of results eliminated in C by each keywords (cost) are:

$benefit(k1) = 4\ (\{R1, R2, R3, R4\})\quad Cost(k1) = 2$

$benefit(k2) = 6\ (\{R5, R6, R7, R8, R9, R10\})\quad Cost(k2) = 6$

$benefit(k3) = 3\ (\{R3, R4, R8\})\quad Cost(k3) = 1$

$benefit(k4) = 4\ (\{R4, R5, R6, R7\})\quad Cost(k4) = 4$

Also if in this example, the set of results in C that is eliminated by a keyword does not intersect with the set eliminated by another keyword.

In this approach, the keywords are always selected in the decreasing order of their benefit-cost values, that is: $k3 \to k1 \to k2 \to k4$ (recall that after a keyword is selected, the benefit/cost of other keywords may change, as discussed in Section 3.1). Having the order of keyword selection fixed, there is a slim chance to achieve the goal of x% elimination. For instance, in order to eliminate 7 results with the fixed order keyword selection, we will have to either use $\{k3, k1\}$ which eliminates 5 results, or $\{k3, k1, k2\}$ eliminating all 10 results. This poses a lot of restriction. Note that in this example, if we do not select keywords in this order, we can choose $\{k1, k4\}$ which eliminates exactly 7 results.

As we can see, always selecting keywords based on their benefit-cost values makes it hard to eliminate a given percentage of the results. Next we discuss the approaches that overcome this problem using a randomized procedure.

3.2.2 Generating queries based on randomly selected subset.

Since selecting keywords in a fixed order is undesirable, this section introduces a randomized procedure. First, a subset of x% of the results in U is randomly selected. Then keywords are selected, aiming at eliminating these randomly selected results. In this way, since the set of results to be eliminated is randomly selected, the keywords will not be selected in fixed order. If the randomly selected set of results is "good", we may be able to eliminate exactly this set of results.

Given the randomly selected results, selecting a set of keywords that eliminate these results with minimal cost is NP-hard, as the weighted set cover problem is a special case of it. To see this, assume that each keyword eliminates part of the selected set of results in U, and their costs are independent (i.e., they eliminates distinct sets of elements in C). Then, each keyword is equivalent to a subset in the weighted set cover problem. To choose a set of keywords that covers the randomly selected results, we can use some greedy approaches, e.g., let S be the randomly selected set of results, at each time we choose a keyword which covers the most number of results in S with minimal cost. Other methods can also be used.

As can be seen here, this approach has two problems. First of all, given a set of randomly selected results, selecting a set of keywords that eliminate exactly this set of results with minimal cost is an NP-hard problem. Second, as illustrated in the above example, the quality of the algorithm highly depends on the selected subset, thus the chance that it can get the optimal answer is still slim.

3.2.3 Generating queries based on randomly selected result.

This section proposes a randomized procedure that has a much better chance to eliminate as close to x% of the results in U as possible. In this method, a result is selected randomly from U that is not yet eliminated, and then a keyword is selected that (1) can eliminate the selected result, (2) and has the highest benefit cost ratio over all such keywords. In case of a tie, the keyword that eliminates fewer results is chosen to minimize the risk of eliminating too many results. The iteration continues until the percentage of results eliminated is smaller than x%.

Example4.2.2.1: Continuing the example, to eliminate all 7 results, we may get the correct solution if we first choose one of the following five results:$\{R1, R2, R5, R6\ or\ R7\}$. Suppose that we choose $R5$, and choose $k4$ to eliminate it. After $k4$ is used, we have the set $R4, R5, R6, R7$ eliminated. Then we can get optimal solution if the next randomly selected result is either $R1\ or\ R2$. To eliminate $R1\ or\ R2$, we choose $k1$, which additionally eliminates results $R1, R2, R3$, totaling 7 results eliminated. As we can see, the approach has a much higher chance to achieve the optimal solution (i.e. removing x% of results) than the one discussed before.

3.2.4 Choosing clusters for result elimination

Note that, given the problem definition in section 2.1, expanded queries are generated independently for each cluster. In each iteration, an expanded query is generated for a cluster, with the goal of maximally retrieving the

results in the cluster and minimally retrieving the results outside the cluster according to goal function in Eq. 4. Given this goal, keywords are chosen to eliminate results from U (outside current cluster C) irrespective of which cluster in U, the result belongs to. It could be possible that with this approach, the algorithm may run into the risk of eliminating all results from a single cluster C' in U and therefore resulting in high overlap between results covered by current expanded query for C and subsequent expanded queries that will be found for clusters U-C. This is especially possible, when the clusters are imperfect and have high overlap of similar results. However, the goal function to be optimized in Eq. 4 is mainly set assuming that the clusters are mostly perfect and may not share many similar results.

In the next section, we will see that this assumption is not always true and learn how to consider the interaction between expanded queries generated in each iteration in the presence of imperfect clusters.

The psuedocode for PEBC is available in the appendix.

CHAPTER 4 ALGORITHMS FOR CQE PROBLEM

As mentioned before, the main goal of this work is to generate expanded queries that are comprehensive in covering all results of the user's query and are also diverse. To achieve this goal, clustering served as a helper tool for generating expanded queries in the earlier approaches.

This section introduces two algorithms, one is Iterative Cluster Refinement (ICR) that minimizes the effect of imperfect clusters and generates quality expanded queries. The other algorithm is Bisecting Query

Generation (BQG) that tries to generate expanded queries directly without the help of clusters.

4.1 ITERATIVE CLUSTER REFINEMENT (ICR)

The algorithms discussed so far, considered retrieving results from other clusters as undesirable and confined the search to finding queries that can retrieve maximal results from its own cluster. In a situation where the clusters are imperfect, this strategy may concede to the imperfectness of the clusters. In case of imperfect clusters, it is very much possible that the optimal queries may cover results from different clusters. In contrast with the earlier approaches, it would be desirable to consider retrieving results from other clusters that are not retrieved by other expanded queries, since we know that the clusters are imperfect and it is baseless to confine the queries to the cluster boundaries. Imperfect clusters are illustrated in the following figure, where the results points are plotted in the keyword space.

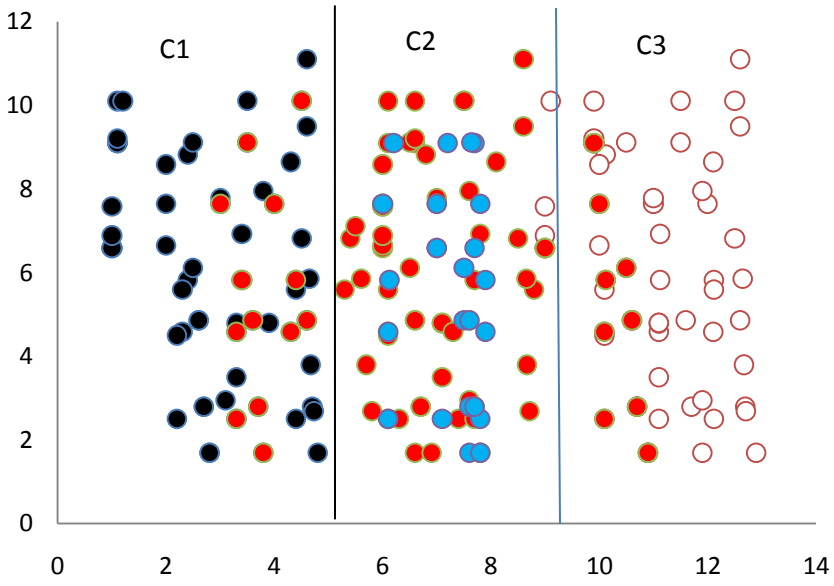


Figure 1: Distribution Of Data Points Of Imperfect Clusters

Two things can be inferred from the above figure: (1) Clustering is not completely bad, it helps in separating highly dissimilar points, example: cluster C1 and C3. (2) However some clusters can be bad, for which the right expanded queries are difficult to generate using approaches discussed in section 4. The experiments reported in section 6.2.3 shows that the cluster quality indeed affects the quality of expanded queries. It can be seen from the experiment results, that even if one cluster has poor quality due to low intra cluster similarity of documents within the cluster and/or high inter cluster similarity with documents in other clusters, the quality of expanded queries tends to be low. Further, it is also seen from these experiments that high quality clusters generally tend to generate quality expanded queries.

The ICR algorithm generates expanded queries by iteratively refining the clusters, so that the effect of imperfect clustering is minimized. Suppose we want to find k expanded queries. The algorithm works in the following steps:

1. Generate an initial set of k expanded queries using either ISKR or PEBC.
2. Pick one expanded query q.
3. Re-cluster the results in $R - R(q)$ into $k-1$ clusters. Generate $k-1$ expanded queries.
4. Repeat from step 2 until k expanded queries are picked.

Coherent with the intuition that the clustering cannot be completely bad, in step 2, the algorithm tries to pick and finalize expanded queries that can correspond to instantly recognizable clusters. In this step, queries that have better alignment with its clusters are picked and made permanent. The strategy for picking these queries is explained below, wherein the basic assumption is that the clusters for which quality expanded queries can be found should be better clusters that may not need much refinement.

In subsequent steps, the results not covered by the finalized query q are considered for re-clustering. This is coherent with the intuition of disregarding the imperfectness seen in the clusters. As the results covered by q are removed, the relationship between the data points might change, and there could be a different set of clusters most relevant to the currently available points after re-clustering.

Choice of queries: Now we discuss how to pick the query in step (2). For the convenience of presentation, I refer to the queries that have been chosen in

step (2) as finalized queries, and the queries that have just been generated in step (3) as candidate queries. First, since the queries already picked are finalized, it is desirable to avoid choosing the queries that overlap with the finalized queries. Besides, a query is a good candidate to be chosen as finalized query, if it has good precision and recall with respect to its own cluster. Therefore, if the F-measure of a query is too low, it should not be chosen as a finalized query.

Based on the above intuition the following desirableness score is used to measure the quality of a candidate query q.

$$Desirableness(q) = p2.\big(1 - overlap(q, Qf)\big) + (1 - p2).Recall(q) \quad (14)$$

Where, Qf is the set of finalized queries, $overlap(q, Qf)$ is the overlap of results retrieved by $q\ and\ Qf$. p is the relative importance of not overlapping with finalized queries, compared to achieving a good F-measure. p is set empirically to 0.7 in the current implementation. A linear combination of overlap and recall is used instead of weighted harmonic mean just to facilitate easier adjustment of weights between 0 and 1.

It would be beneficial to learn the value of "p" based on observed intra cluster and inter cluster similarity over a range of training queries. The value of p can be set based on a threshold value of the lowest intra cluster similarity Th_{Intra} and the threshold value of highest inter cluster similarity Th_{Inter} among the clusters. Since the values of these internal measures may largely depend on the underlying data, it is not reasonable to use a static function of these measures to compute the value of "p". Rather, the threshold

values must be learned over a set of training queries on the underlying data in order to deduce the value of "p".

Generating Candidate Queries:

In step 3, the algorithm re-clusters the remaining uncovered results and generates expanded queries using PEBC or ISKR. Note that while selecting keywords to be added to the expanded queries using PEBC/ISKR, extra weight can be given to keywords that do not retrieve results that are already covered by the finalized queries. The benefit factor computed in ISKR/PEBC while picking keywords is changed to accommodate this weight:

$$benefit(k,q) = S\big(R(q) \cap Uc \cap E(k)\big) + p2.S(R(q) \cap Uf \cap E(k)) \quad (15)$$

Uc is the set o results in other clusters, which is the same as U in Equation 9. Uf is the reasults covered by the finalized queries. p2>1 is the importance weight for not retrieving such results. We empirically set p2=3. The cost factor remains the same, and the value of the keyword is still computed as ratio of benefit and cost.

ICR thus tries to improve the quality of expanded queries by taking an aggressive approach to improve coverage, and a cautious approach to maintain low overlap. It tries to be aggressive in covering results not retrieved by other expanded queries even though they may correspond to other clusters, at the same time it tends to be cautious in not retrieving results already covered by finalized queries. The experiment results show that

this methodology significantly improves upon the quality of expanded queries generated by ISKR and PEBC.

The following example illustrates the steps in ICR algorithm:

Example 5.1.1: Consider the following clusters obtained for the user's query "Columbia" and the results covered by some of the keywords:

$C1: \{R1, R2, R3, R4, R5\}$

$C2: \{R6, R7, R8\}$

$C3: \{R9, R10\}$

album $- \{R1, R2\}$

city $- \{R1, R2, R4, R5, R6, R7, R8\}$

district $- \{R4, R5, R6, R7\}$

indiana $- \{\mathrm{R7}, \mathrm{R8}\}$

university $- \{R9, R10\}$

The algorithm first processes the clusters and picks {university}, {Indiana} and {album} as the expanded queries for corresponding clusters. Given the three clusters, these are the best queries that can be obtained by the approaches discussed in section 4. As can be seen, many results are left uncovered by the above expanded queries. ICR picks "university" and finalizes it as one of the expanded queries. It then takes the rest of the uncovered results and re-clusters them. Let's suppose the clusters obtained by re-clustering are C1: {R3, R4, R5, R6, R7}, C2: {R8, R1, R2}. ICR picks "district" as the next finalized query, which leaves out only one cluster of uncovered results, {R3, R8, R1, R2}, ICR finally outputs the final set of

expanded queries as, {university}, {district} and {album}. As can be seen, these queries have better coverage than the previous set of queries.

4.2 BISECTING QUERY GENERATION (BQG)

Although ICR tries to minimize the effect of bad clusters on expanded queries, clustering always incurs some amount of loss on the quality of expanded queries. This section discusses an algorithm that tries to directly generate expanded queries without the help of clusters.

The problem of query expansion can be seen as feature selection problem, where the query results are represented in N dimensional space of keywords/terms. Now the problem is to find a subset of diverse term dimensions that can represent the documents comprehensively. Various feature selection methods such as term frequency based [Cutting et al. 1992], centroid [Radev et al. 2004], document frequency based or mutual information based [Manning et al. 2008] can be applied to solve this problem. However, all these methods suffer from the various problems of cluster labeling methods mentioned in section 1.

Closely connected with works on mapping multimedia objects in N dimensional space to K-d space [Faloutsos et al. 1995, Torgerson 1952], this section presents an algorithm named as Bisecting Query Generation (BQG) that exploits the information about distances between query results on the basis of term composition to directly find K expanded queries.

The BQG algorithm proceeds in following steps:

1. Generate initially two single keyword queries q1={k1} and q2={k2}, such that among all single keyword queries, q1 and q2 has the maximum score per equation 8.
2. If the number of queries has reached the threshold K, terminate. Otherwise:
    a. Find another expanded query, If the F-measure score as per equation 8 can be improved by doing so; Else:
    b. Pick one of the already found expanded query q, split q into two queries q' and q'', each having one additional keyword than q, such that these two queries maximize the F-measure score for the results retrieved by q.

In each iteration step 1 and 2(a) tries to find orthogonal term dimensions that can improvise the F-measure score until K dimensions of expanded queries are found. Considering the result documents are points in N dimensional space of terms, step 2(b) considers a (N-1) dimensional hyper-plane H, and continues to find orthogonal term dimensions within this hyper-plane until K dimensions are found. The strategy for picking this hyper-plane H is explained below, the basic intuition is to pick the one that can be further naturally divided into more orthogonal dimensions.

Choosing the query to refine:

There are several ways to choose the query for further refinement in step 2(b). For example: (1) Picking the query based on least number of keywords; (2) Picking the query which has most number of results. However these approaches may not work in many cases, because different categorization of

the original query may have different number of results, and may be best described by different number of keywords. For example, when we search "java", the two initial queries may be "java, language" and "java, location". Although they both have the same number of keywords, and "java, language" retrieves much more results than "java, location", yet it might be desirable to refine query "java, location" since "java" matches many locations, e.g., an island in Indonesia, a town in Georgia, etc.

As we can see, which query we use to refine the results should not depend on the number of keywords or the number of results it retrieves, but on whether its results can be naturally divided into multiple categories. Therefore, to decide a query to refine, the query whose results have the minimum average similarity is selected. The standard metric in IR, the cosine similarity of two vector is used as the similarity measure of two results. If the results are text documents, each component of the vector is a keyword and its value is the TF of the keyword. If the results are structured documents, each component is a feature, and the value is the TF of the feature. The desirableness of choosing query q to refine is defined as:

$$Desirableness(q) = -\sum_{r,r' \epsilon\, R(q)} Sim(r, r') \qquad (16)$$

To compute similarity of every pair of documents in the result set of q would be expensive. One heuristic alternative is to find the centroid of the result set and find the average similarity of all documents with the centroid. Also note that, In step 1 and 2(b), finding best pair of keywords that can maximize the

F-measure is a N^2 operation, where N is specifically the number of dimensions or terms in the result set. However feature reduction strategies such as considering only keywords with high term frequency and/or high document frequency can be applied to improve the efficiency. In the current implementation, keywords that appear in at least two documents with a term frequency of 3 are chosen, based on best results obtained compared with other strategies.

The following example illustrates the working of BQG algorithm.

Example 5.2.1

Given below are some of the keywords and their result coverage of the results of original query "Columbia".

"Indiana" - {R1, R2, R3, R4}

"Album" - {R1, R5, R6}

"City"- {R3}

"District"- {R3, R4}

"University"- {R2, R6}

The ICR algorithm can generate good expanded queries if in each iteration, at least one expanded query is good (i.e., at least one cluster is good) and this query is the one selected to be the finalized query. Therefore, it is still dependent on the quality of clustering to certain extent. On the other hand, the BQG algorithm completely eliminates the dependency on clustering by generating two initial queries, and splitting one queries into two at each step. The BQG algorithm first enumerates every pair of keywords and find the pair with the highest score per Eq. 8. Among these six keywords, "Indiana" and

"Album" have the highest scores: queries {"Indiana"} and {"Album"} has a coverage of 1 and an overlap of 0.33, thus their score is the harmonic mean of 1 and 0.67, i.e., 0.80. The current expanded queries are thus q1 = {"Indiana"} and q2 = {"Album"}.

Now, we pick a query to split. Among the current expanded queries, we choose the one whose result has the minimum average similarity, which is "Indiana". Again, we enumerate every pair of keywords and find the pair of keywords k and k', such that queries {"Indiana", k} and {"Indiana", k'} have the largest score with respect to the results retrieved by query {"Indiana"}. The best pair of keywords are "University" and "District", since {"Indiana", "University"} and {"Indiana", "District"} have good coverage and zero overlap. Now we have three queries with full coverage and no overlap. Therefore, the BQG algorithm will output: q1={"Album"}, q2={"Indiana", "University"} and q3={"Indiana", "District"}.

CHAPTER 5 EXPERIMENTS

In this section, a set of experimental evaluations are presented on the quality of expanded queries generated by the current approach, and the efficiency and scalability of query generation.

5.1 EXPERIMENT SETUP

Environment: All experiments were performed on a machine with AMD Athlon 64 X2 Dual Core Processor 6000+ CPU with 3GHz, 4GB RAM, running Windows Server 2008.

Dataset: Two datasets are used for evaluation: shopping and Wikipedia. Shopping is a data set that contains information of electronic products crawled from circuitcity.com. Each product has a title, a category, and a set of features. Wikipedia is a collection of document-centric XML files used in INEX 2009.

Query Set: 10-12 queries are tested on each data set, as shown in Appendix A. The queries on Wikipedia dataset are composed of ambiguous words. The queries on shopping dataset are to search for specific products.

Result Clustering: Each result is modeled as a vector whose components are features in the results and the weight of each component is the TF of the feature. The similarity of two results is the cosine similarity of the vectors. Feature reduction is applied to reduce the number of dimensions for improving efficiency, specifically only the terms with document frequency of 2 and above and term frequency of 3 and above are selected. The clustering algorithm used is K means with varying-K approach, for determining the number of clusters dynamically. The algorithm iteratively generates clusters with incremental K values and stops when rate of change of goal function flattens. The goal function is set as benefit-cost, where benefit is the sum of intra cluster similarity of documents with their respective centroids and cost is the weighted function of the number of clusters [Manning et al. 2008].

Comparison System: Following are some of the search systems providing query expansion service on which the test queries are evaluated for comparison.

(1) Data Clouds [Koutrika et al. 2009], which takes a set of ranked results, and returns the top-k important words in the results. The importance of a word is measured by its term frequency in the results it appears, inverse document frequency, as well as the ranking score of the results that contain the word. Data Clouds is a representative method for returning important words in the search results, without clustering the results.

(2) Google. For each test query, the first 3-5 related queries suggested by Google (the number of which is the same as the number of queries generated by other approaches) are chosen. Google is a representative work of suggesting related queries using query logs.

(3) F-measure, which is an alternative ISKR algorithm that considers the value of a keyword k with respect to a query q as the delta F-measure of q after adding k to q or removing k from q. As discussed in Section 4, since the goal function is to maximize the F-measure of a query, the delta F-measure more accurately reflects the value of a keyword than the benefit-cost values. However, in this approach, after a keyword is added to or removed from the current query, the values of all keywords will need to be updated, which potentially leads to a low efficiency.

(3) TFICF, a frequency based feature selection method representing term frequency based Cluster Summarization [Carmel et al. 2009]. It first clusters the results, then generates a label for each cluster. The label of a cluster is selected based on the term frequency (tf) and inverse cluster frequency (icf) of the words in the cluster. This is a representative method for cluster summarization and labeling.

(5) Mutual Information based feature selection [Carmel et al. 2009, Manning et al. 2008], selects terms for each cluster based on the measure of how much information – in the information theoretic sense – a term contains about the corresponding cluster. MI reaches its maximum value if the term is a perfect indicator for cluster membership, that is, if the term is present in a result if and only if the result is in the cluster.

(6) Chi Square based feature selection [Tseng et al. 2006, Liu et al. 2003, Carmel et al. 2009, Manning et al. 2008, Yang et al. 1997], where terms are selected for each cluster based on the measure of independence between the occurrence of the terms and the occurrence of corresponding cluster. Since, Chi square considers both positive and negative correlation of terms with clusters and may tend to output some negative terms that are indicative of non-membership in the cluster, square root of Chi square is considered which is nothing but "correlation co-efficient" [Tseng et al. 2006]. It outputs only the positive terms that are highly indicative of membership in a class.

(7) The algorithms discussed in the current work namely, ISKR, PEBC, ICR and BQG are evaluated and compared with the above systems.

For both the datasets, all systems consider top 100 results to generate expanded queries.

5.2 QUALITY OF QUERY EXPANSION

5.2.1 User Study.

An extensive user study on Amazon Mechanical Turk [1] was performed with 50 public web user participating in the study. The user study consists of three

parts. In the first part, the users gave ratings to each individual expanded query, in the second part the users rated the queries considering collectively the set of expanded queries generated by a search system. In the third part, a general question was asked to the web users in order to verify the intuition of the approaches.

Part 1: Individual Query Score.

In the first part of the user study, the users were asked to rate each individual expanded query in a 1(low)-5(high) scale, based on how they feel about the expanded queries. The users were also asked to choose the following justification options to reason their ratings.

A – The expanded query is highly related to the search and helpful.

B – The expanded query is related to the search, but there are better ones.

C – The expanded query is not related to the search.

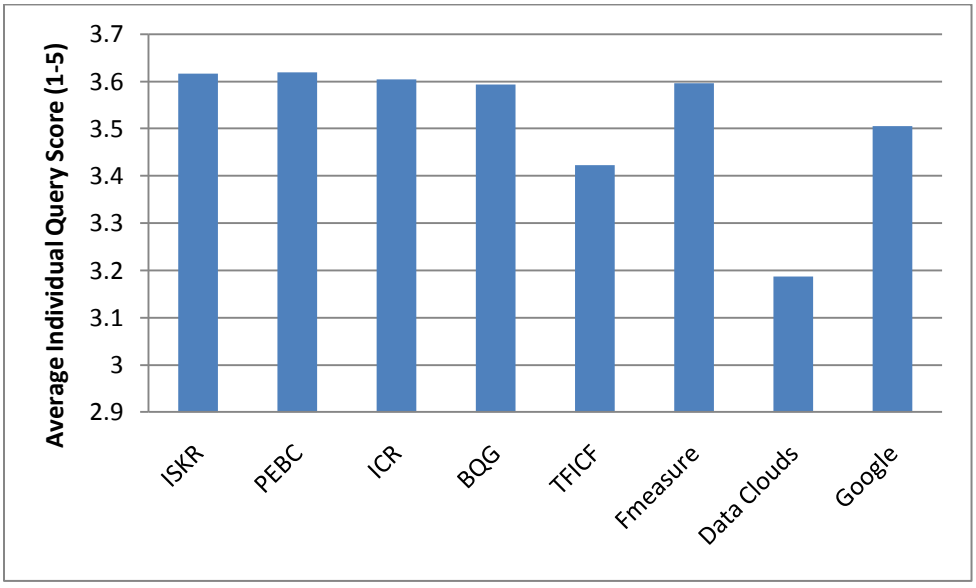


Figure 2: Average Individual Query Score

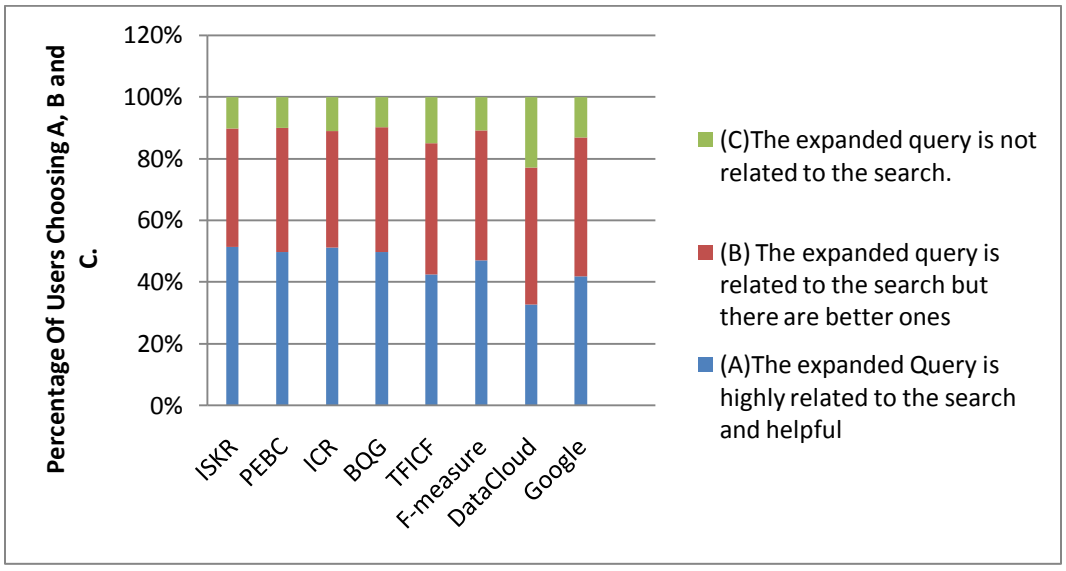


Figure 3 : Percentage Of Users Choosing A, B, or C For Individual Queries

The average score of all 22 queries given by all users for each approach is shown in Figure 3, and the percentage of users choosing each option in this part of the user study is shown in Figure 4. As can be observed, the current work's approach is rated better than other approaches. Also as can be seen from figure 4, many users found the current work's approaches produce highly related and helpful expanded queries compared to other approaches.

BQG iteratively finds terms that can cover more results of the original query and has less overlap with already discovered terms. In general users have rated individual queries of BQG on par with other cluster based approaches. For some individual expanded queries (For example: Original queries QW5 "Eclipse", QW9 "Mouse" etc) which are discovered at the last just before the iteration ends, the users have given low rating. One reason for this trend could be that, these queries/terms make lesser semantic sense to

the users as they cover less number of results in the cluster although adding such terms to the expanded query set improves the goal function. This situation can be handled by carefully selecting the condition to stop the iteration. In the current implementation, various factors lead to the termination of the iteration, such as the upper bound on the number of expanded queries, the non-availability of any terms to further improve the goal function, the threshold on rate of increase in the goal function etc. These factors are hard to be set arbitrarily, and rather needs to be set by periodic learning and techniques such as relevance feedback.

The TFICF approach chooses keywords that are popular in the current cluster in contrast with other clusters, but however it does not considers the interaction between the keywords featured in an expanded query, thus may tend to pick keywords hat have high occurrence (TF), but with low co-occurrence in the cluster results. Thus the users mostly found such expanded queries less desirable. For example, for query "Jaguar", TFICF approach generated expanded query "Jaguar, OS, nova", for query "Rockets", it generated "Rockets, games, artillery" etc.

Google chooses keywords based on query log, thus it often returns meaningful and popular keywords, which is desirable. For example, for QW6 "Java", Google returns the expanded queries "Java, Tutorials", "Java, Games" etc., which are generally very popular with the users. However, for some queries Google may return keywords that do not occur in the results. For example, Consider QS1 "Canon, products", Google returns a query "Olympus products", "Nikon products" etc. While this could be useful for some users,

the user rating has indicated that many the users prefer the expanded queries to be results oriented.

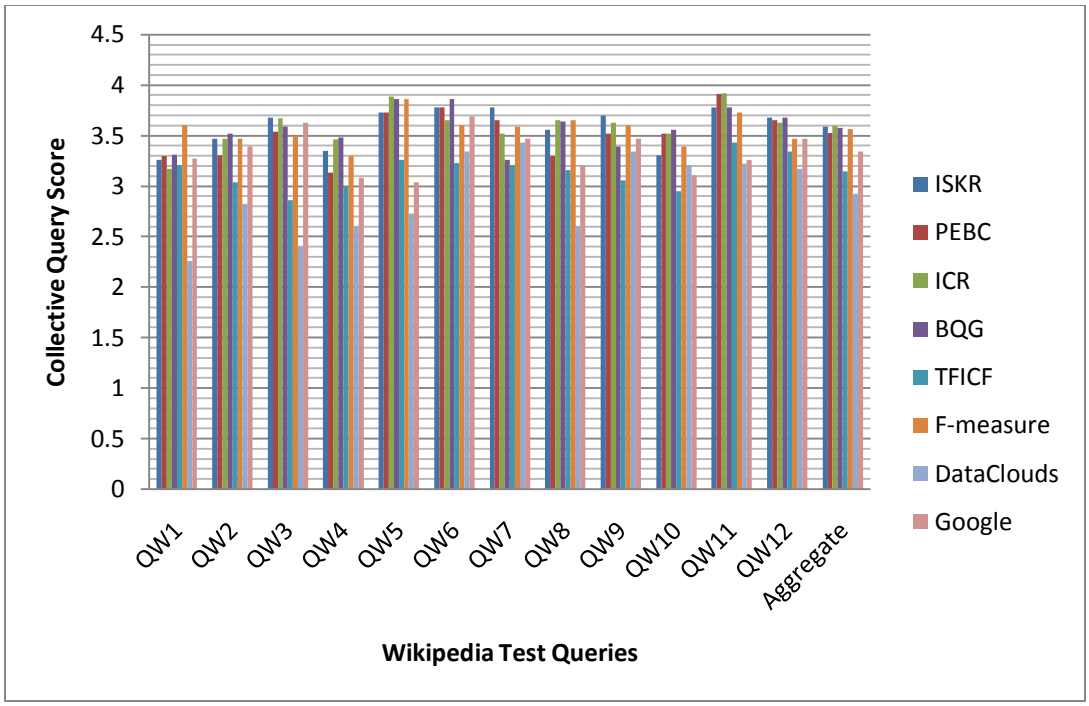
Part 2: Collective Query Score.

In the second part of the user study, the users were asked to rate each set of expanded queries generated for a user query in a 1-5 scale, based on how they feel about the collective set of expanded queries returned by a search system given a original query. The users were also asked to choose from the following justification options:

A - Not comprehensive and not diverse.

B - Either not comprehensive or not diverse.

C - Comprehensive and diverse.



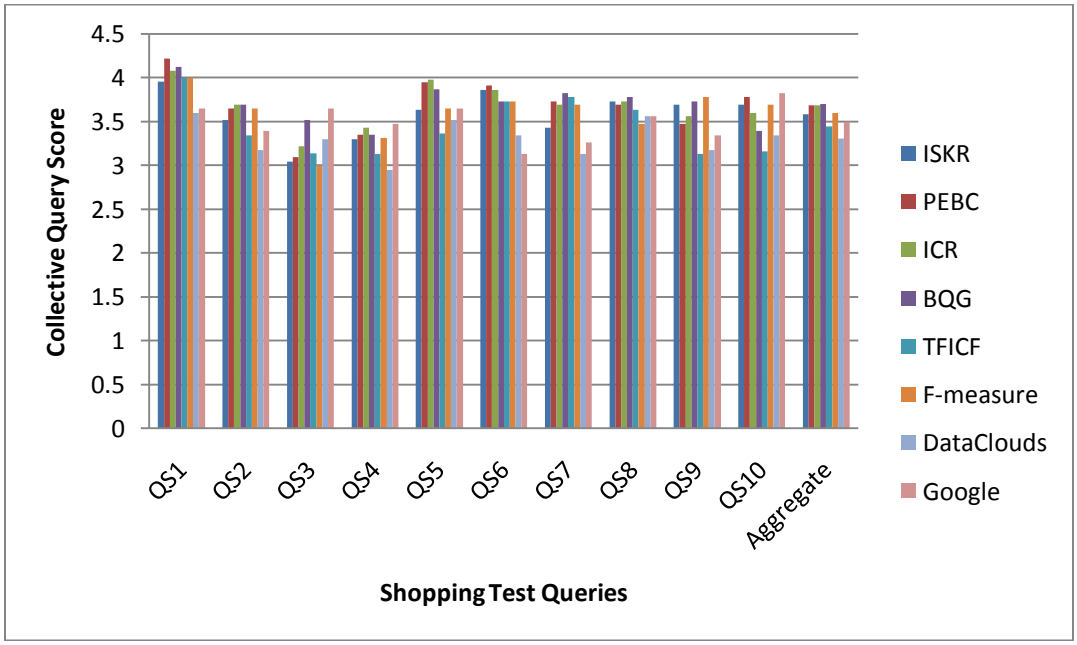


Figure 4: Collective query scores for each set of expanded queries

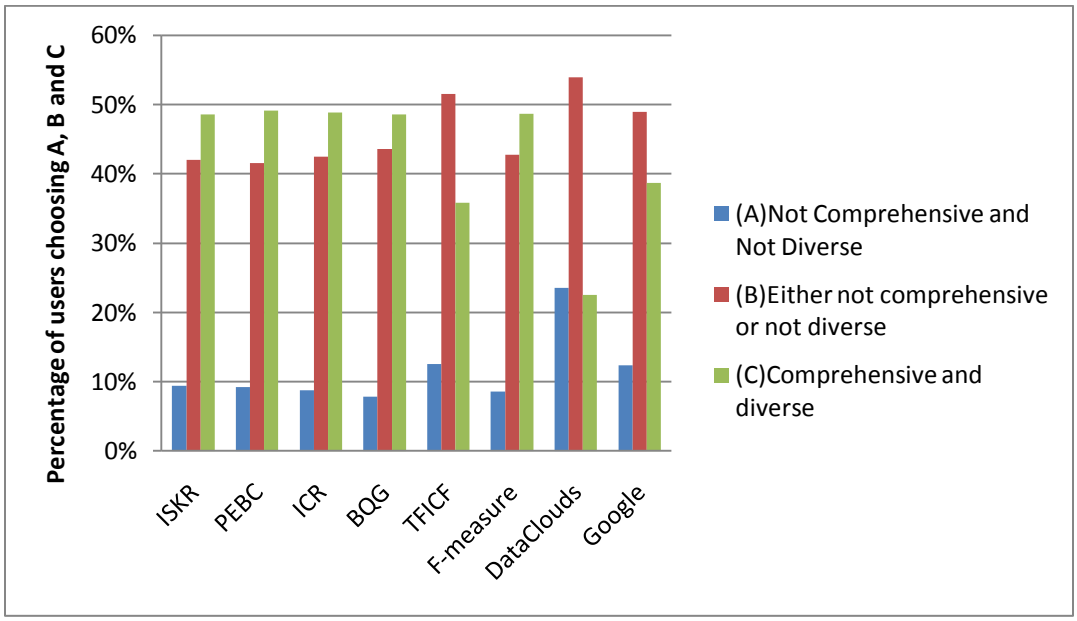


Figure 5: Percentage of users choosing A, B, C for each set of expanded queries

For all 20 queries, the collective score of each user query for each approach is shown in Figure 5, and the percentage of users that chose each option is shown in Figure 6.

For may queries such as, "Domino", most of the current work's approaches generated expanded queries covering comprehensive and diverse topics such as For query QW4 "Domino": "pizza", "album", "company", "lotus notes database" etc.; For QW5 "eclipse": "software", "solar eclipse", "car", "album"; For "Java": "Indonesia", "Software"; For QW10 "Jaguar": "animal", "car", "apple system" etc. All such queries gained higher ratings from the users when compared to other approaches. Since the "Frequency " approach also aims at finding keywords that differentiate clusters, the users found TFICF generate diverse expanded queries for many test queries. For some of

the queries such as "Eclipse", TFICF found terms that occurred in fewer results with high frequency, such as "Eclipse astronomical mathematics". The users should have found such queries too specific to serve as a label for a broad topic related to Eclipse. For some cases, where the clusters are noisy TFICF is not found to be comprehensive and diverse, such as for the query "Domino", "Eclipse" etc. For such cases, ISKR and PEBC also received relatively lower rating compared to ICR and BQG. ICR and BQG succeed at covering comprehensively all topics maintaining the diversity even in the presence of noisy clusters, and therefore received better collective ratings from users.

Data cloud returns the top ranked keywords in the results for which the expanded queries often lack comprehensiveness and diversity. For example, consider QS1 "Canon, products". All the current work's approaches return camera, printer and camcorder. However, Data Clouds returns all expanded queries related to camcorders, as there are many results that correspond to camcorders. The users mainly chose option A or option B for Data Clouds.

For many queries in Shopping data such as "HP Products", "Canon Products", "Memory" etc., TFICF received better ratings on par with other approaches. This is because the shopping data is more structured and results in the same cluster are highly coherent and share many common features. Therefore, even though the TFICF approach does not consider the relationship of keywords, the keywords it selects in an expanded query likely co-occur in many results. On the other hand, on the Wikipedia data, it may choose a set

of keywords, such that each of them has a high occurrence but they do not necessarily co-occur. Such a query will not retrieve many results which lowers its recall.

Part 3: User opinion about expanded queries.

The users were asked about their general opinion about a good set of expanded queries, in order to gain an understanding of whether the assumptions made in the current work are aligned with the need of a web user. Following are some of the responses received:

"A best expanded query has the power to decide what the user wants."

"The expanded query should be short and precise."

"A good expanded query should be specific enough for what the person is looking for but also general enough so that it doesn't get too specific."

"It should contain different areas of relations to the searched words/phrases."

"Comprehensive, useful, with options."

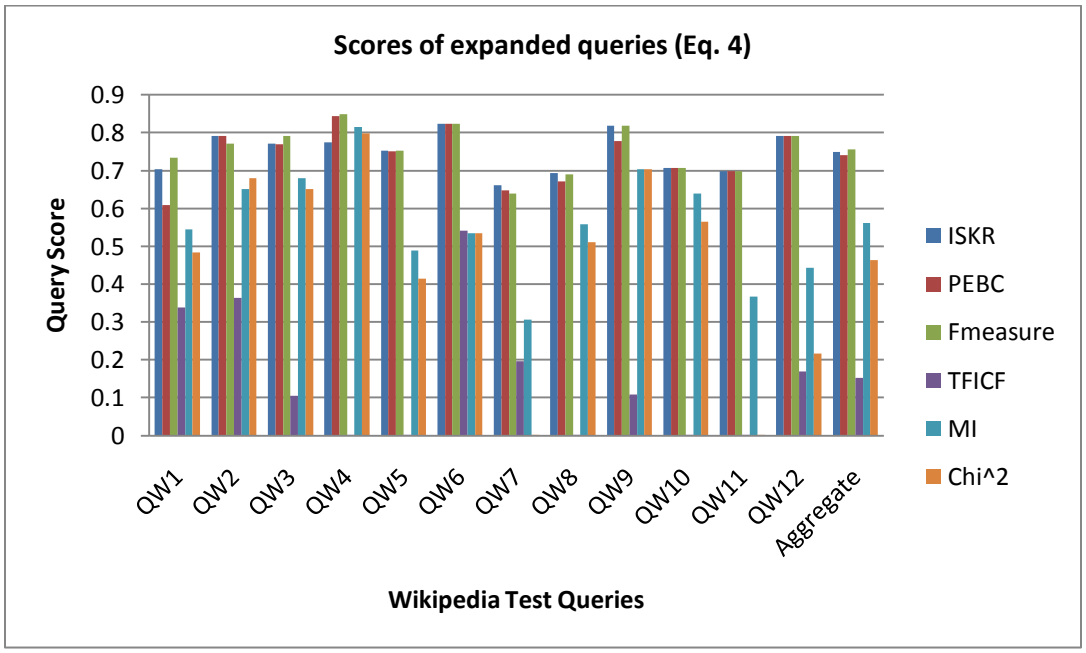
"Need some of the options as different from other options."

These responses mostly indicate comprehensiveness and diversity as desirable features.

5.2.2 Scores of expanded queries (using eq. 4).

As defined in Eq. 4, the score (goal function) of a set of expanded queries is the harmonic mean of their F-measures. In this section, the scores of expanded queries of ISKR, PEBC and TFICF approaches are shown in the figure 7. Since the queries generated by Data Clouds and Google are not

based on clusters, this score is not applicable to them. The ICR and BQG approaches are not comparable here as they are evaluated using a different score measure given in Eq. 8 and moreover all the approaches tested in this part consider clusters as ground truth and use Eq. 4 as the evaluation measure. The comparison with the ICR approach is provided in the next section.



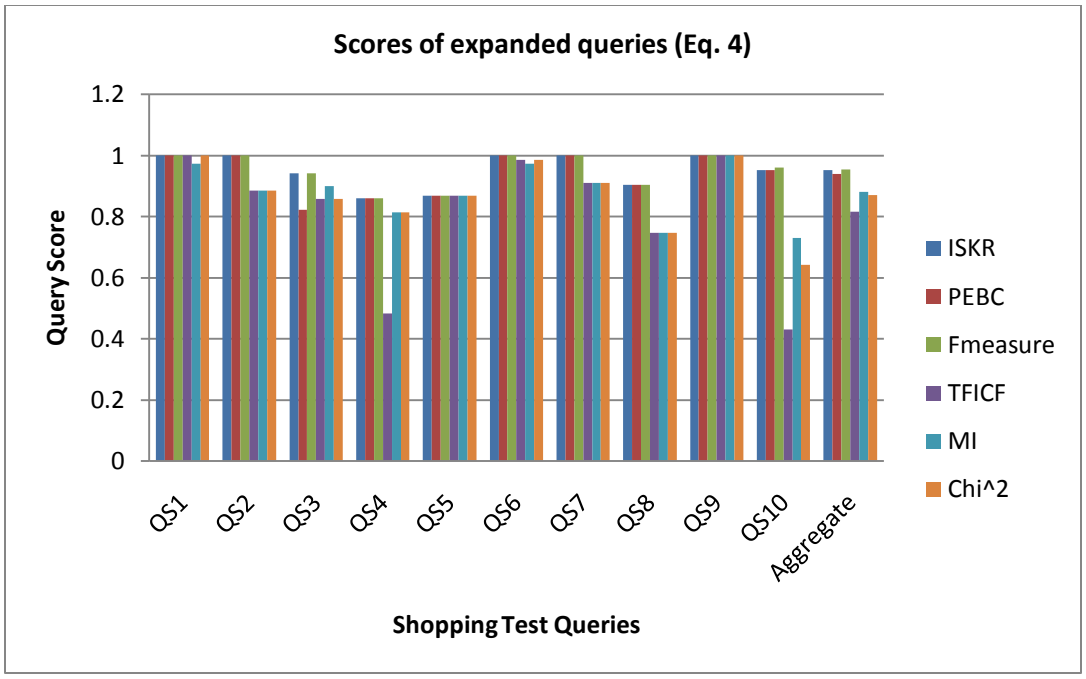


Figure 6: Scores of expanded queries considering clusters as ground truth (Eq. 4)

As we can see, in general both ISKR and PEBC algorithms achieve similar and good scores. On the shopping data, both algorithms achieve perfect score for many queries. This is because on the shopping data, products of different categories usually have different features. Thus for queries whose results contain several different product categories (e.g., QS1 "Canon" whose products contain camcorders, printers, and camera), each category forms a cluster. Therefore, it is usually possible to achieve a perfect precision and recall.

The TFICF based approach as discussed before selects keywords having high frequency, but does not takes into account the number of results in which the keywords are present and their co-occurrence in the data. As a

result they have poor scores for the wiki dataset. Since in shopping data, many results in the shopping data usually have common keywords, TFICF tends to have better scores for shopping data.

5.2.3 Effect of cluster quality on expanded queries.

The main motivation for the algorithms described in section 5 is to minimize the effect of bad clusters on the quality of expanded queries. However, the precise relationship between cluster quality and quality of expanded queries is not yet discussed. This section studies the effect of cluster quality on the generation of quality expanded queries. Specifically since the algorithms in section 4 aims to generate expanded queries which cover maximal results from the same cluster and minimal results outside the cluster, we will see in this section, how even one bad cluster may affect the overall score of the expanded queries. Internal cluster evaluation measures [Manning et al. 2008] such as Intra cluster Similarity and Inter Cluster Similarity are used to determine the quality of the clusters.

Intra cluster similarity of a cluster is the average similarity of document results within the cluster, Inter cluster similarity between two clusters is the average similarity between document results of the two different clusters.

$$IntraClusterSim(Ci) = AVG(\bigvee_{r,r' \epsilon Ci} Sim(r, r')) \quad (16)$$

$$InterClusterSim(Ci, Cj) = AVG(\bigvee_{ri \epsilon Ci, rj \epsilon Cj} Sim(ri, rj)) \quad (17)$$

For each of the test queries, the lowest intra cluster similarity and the highest inter cluster similarity values found among the clusters is compared with the overall score of expanded queries in Figure 8. Both these internal measures indicate the level of poor quality of the clusters.

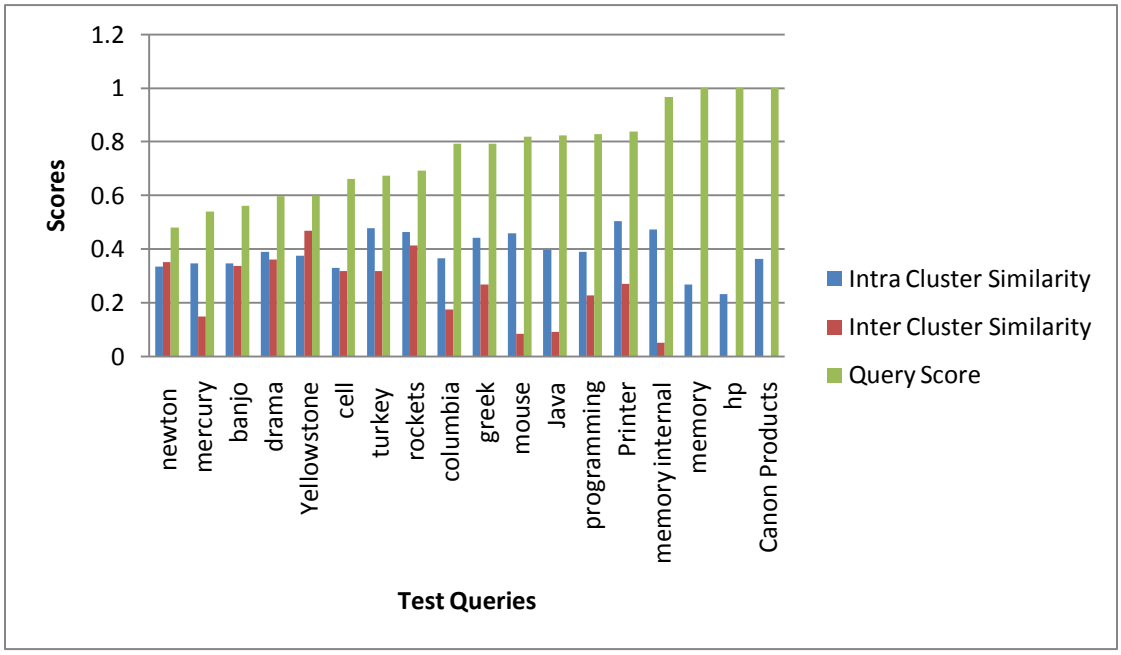


Figure 7 Effect Of Cluster Quality On Expanded Query Score

As can be observed from the above figure, high inter cluster similarity generally results in low expanded query scores. Intuitively, this is plausible as it becomes increasingly difficult for the algorithms to find terms that can eliminate maximal results from other clusters when the clusters are similar. The above figure shows a general trend of inverse correlation between inter-cluster similarity and query score.

In case of intra cluster similarity, high intra cluster similarity in a cluster should intuitively help the algorithms to find expanded query terms that can maximally retrieve results from the cluster. This can be observed by looking at the query "newton", "turkey" and "rockets". For query "rockets", the inter-cluster similarity is higher than "newton", but yet with the support of high intra cluster similarity, the algorithm generates high query score. The same phenomenon can be observed when comparing the query "turkey" and "newton". However, sometimes when the intra-cluster similarity is low, i.e, when results in the same cluster share less number of terms, the algorithm still manages to find a few shared terms that can cover the cluster well, for example: query "hp", "memory" etc. In general, it is desirable to have high intra cluster similarity and low inter cluster similarity to get better quality expanded queries.

Therefore, these results support our motivation to negate the effect of poor clusters for generating better expanded queries. However, in order to substantiate this intuition, I did the following case study to find out how the expanded queries look semantically when the cluster quality is high/low. Specifically, this case study helps in understanding whether a semantically better query actually corresponds to a high f-measure score. The semantic meanings are cross verified with Wikipedia's disambiguation pages [28] for wiki queries and with the help of domain knowledge for Shopping queries.

Case Study Example 1: Query "Mercury" has lot of results mainly about albums and magazines of the name Mercury. ISKR generates expanded queries "Mercury, center", "Mercury, album" and "Mercury, century". Thus

ISKR is able to identify only one main cluster in its expanded queries. This is because except for one cluster, the other clusters are impure with a mix of results from various topics like "Magazines", "Cars", "TV Series" and "news articles about planets" etc. ISKR does not selects "magazine" as one of the expanded keywords because it tries to avoid overlap with other cluster which also contains results about journals and magazines. Therefore, the expanded queries tend to have low recall and therefore low f-measure. The internal evaluation measure also indicates poor quality for these clusters.

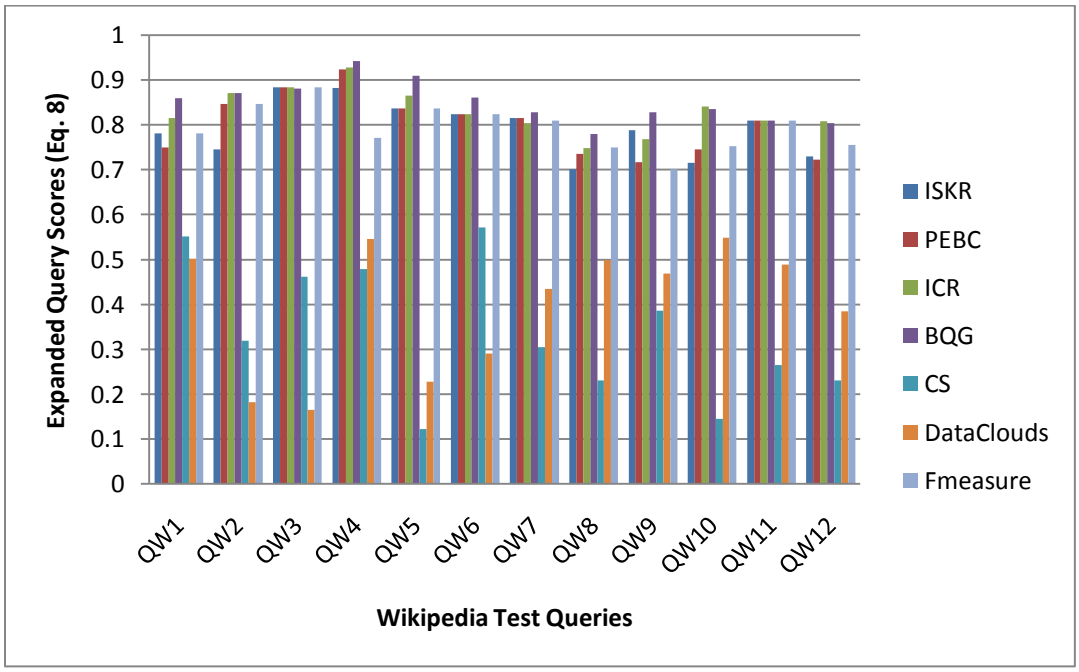
Case Study Example 2: Query "Banjo" also suffers from similar problem, as a majority of result distribution corresponds to the topic "Guitar and albums". Other results do not correspond to any bigger classification, as a result only one expanded query generated by ISKR looks meaningful ("Banjo, album"). Query "Yellowstone" has about 80% of result distribution corresponding to "Parks", as a result its clusters have high inter cluster similarity, subsequently ISKR generates two expanded queries having the keyword "Park" such as "Yellowstone, glacier, park" and "Yellowstone, park, pass". As a result it fails to identify topics corresponding to smaller distributions such as "Montana county", "volcano" etc.

Case study example 3: Pure clusters whose internal measure quality is high generally resulted in semantically meaningful expanded queries. For example: Test queries like "Canon Products", "HP", "Memory" etc. have meager inter cluster similarity and ISKR is able to find optimal classifications for these queries, for example: For "Memory", ISKR generates expanded queries

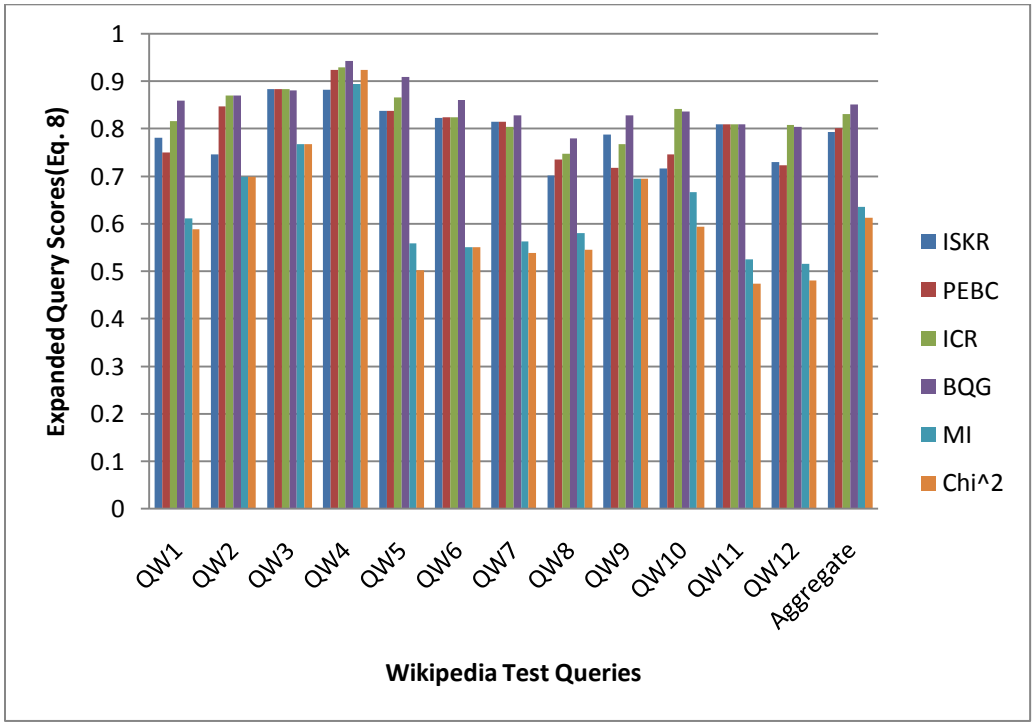
"ddr3", "flash memory" and "hard drive", which are meaningful and correspond to distinct classifications.

5.2.4 Scores of expanded queries (using eq. 8).

In this section, experiment results of comparing the ICR and BQG approaches with other approaches are presented. Since ICR ad BQG needs to be evaluated based on a different score given in Eq. 8, all approaches are evaluated for the same score and compared with each other.




1.2
1
0.8
0.6
0.4
0.2
0
Expanded Query Scores(Eq. 8)
QS1
QS2
QS3
QS4
QS5
QS6
QS7
QS8
QS9
QS10
Shopping Test Queries
ISKR
PEBC
ICR
BQG
CS
DataClouds
Fmeasure


1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
Expanded Query Scores(Eq. 8)
QW1
QW2
QW3
QW4
QW5
QW6
QW7
QW8
QW9
QW10
QW11
QW12
Aggregate
Wikipedia Test Queries
ISKR
PEBC
ICR
BQG
MI
Chi^2

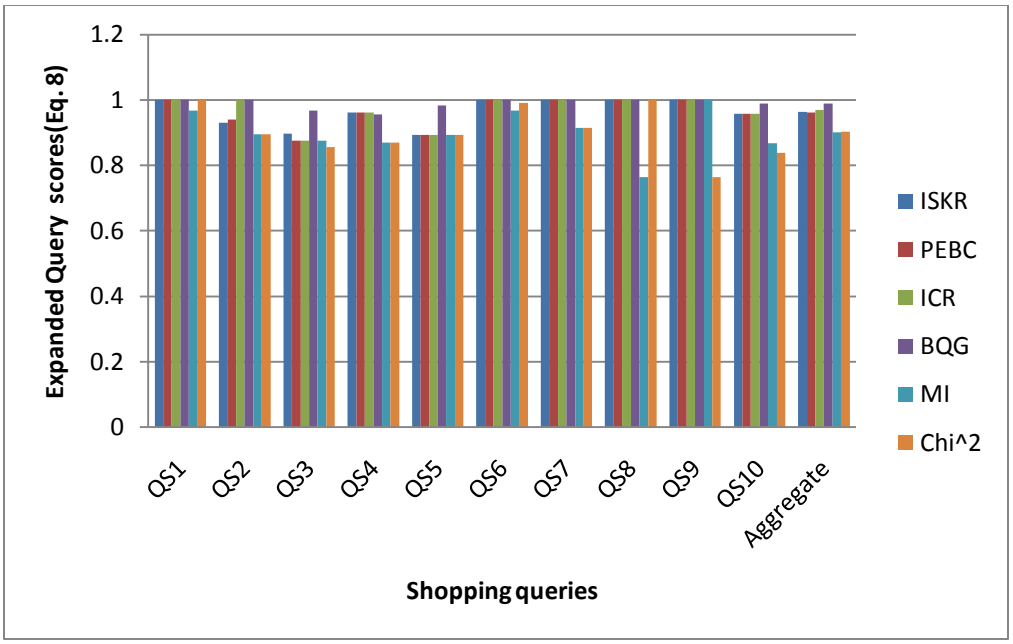


Figure 8: Scores of expanded queries (Eq. 8)

In general the Mutual information and Chi^2 feature selection methods generate meaningful expanded queries with terms representing one cluster and contrasting other clusters. One problem with these methods is that, they consider terms independently without considering the effect of adding a term to an expanded query on the optimization goal. As a result, they tend to pick terms for an expanded query that independently may have high occurrence in a cluster, but together as an expanded query have less co-occurrence, resulting in poor recall. For example: For the original query QW8 "Cell", one of the expanded queries generated by MI is "Cell, wall, table", resulting in recall of 0%. Semantically, it is true that "Cell, wall" and "Cell, table" are completely different topics, and this query is not useful as a categorizing expanded query for query refinement. Another example of this problem is for

QW5 "Eclipse", one of the expanded queries generated by Chi^2 is "Eclipse, solar, fiction", which has a low recall. The overall score of almost all the expanded queries is low for MI and Chi^2 approaches when compared to the current work's approaches.

The expanded queries of each approach corresponding to the above scores are provided in Appendix D. Given below is detailed case study analysis on the results:

Case Study Example 1 – Query "San Jose", many results of San Jose are about San Jose Location, with a very small group of results about local NHL team based on San Jose. Clustering dependent approaches do not identify this classification, and form clusters which are both about location. As a result the cluster labels are not diverse ("San Jose, California", "San Jose, province"). Whereas, ICR and BQG are able to identify this classification ("San Jose, Santa", "San Jose, team", "San Jose, players" and "San Jose, California" etc.). According to cluster metrics, the average Intra cluster similarity of documents is around 0.48, and the average inter cluster similarity is 0.30, which indicates that the clusters share a lot of common keywords and therefore have a lot of overlap. Therefore ISKR and PEBC approaches find it difficult to extract words having high f-measure; the expanded queries generated by these approaches have high overlap of 31%. ICR is able to identify diverse classifications of "team" and "location", which reduces the overlap to about 6% maintaining same coverage of 81% as other approaches, and therefore resulting in high f-measure score.

Case Study Example 2 – Query "Domino", although most of the queries generated by ISKR and PEBC are meaningful and corresponds to main classifications of the results such as "domino, album", "domino database", "domino products", they still miss at least one classification due to formation of a bad cluster in the presence of outliers. For example, one of the queries generated by PEBC is about "domino, California, border" etc. This does not correspond to any meaningful classification related to the query "Domino". The cluster corresponding to these expanded queries have very few results and is mainly formed due to some outliers selected as initial centroids.

ICR helps in eliminating the effect of such bad clusters formed due to outliers, and extracts meaningful classifications such as "Domino pizza", which was missed out by naturally formed clusters.

Case Study Example 3 – Query "Rockets", TFICF can sometimes generate queries that can belong to two different topics in the same query. This is because; clusters can be bad and may contain more than one topic. And also since TFICF just outputs most frequent terms in the whole cluster and does not considers their interaction(results retrieved), it generates keywords from multiple topics in the same cluster. This can be misleading to the users. For example: For Rockets, one of the queries by TFICF is "Rockets, games, artillery" which does not make much sense as "artillery" and "games" do not retrieve any results together.

Case Study Example 4 – Query "Jaguar", ISKR and PEBC approaches, although they generate meaningful expanded queries like "Jaguar, tiger", "Jaguar, car" etc., due to imperfect clusters, the queries are not highly diverse. For example: Since there are many results about "Jaguar Car", two of the clusters are very close to each other in terms of inter cluster similarity sharing results about Jaguar cars. Since ISKR and PEBC aim to cover maximal results from the same clusters and minimal results from other cluster, they end up generating queries which either have low coverage and low overlap or high coverage and high overlap in the presence of imperfect cluster, where satisfying both the desired criteria is difficult. In this case, PEBC generates expanded queries "Jaguar, Car" and "Jaguar, Season" which has moderate coverage and high overlap, whereas ISKR generates queries "Jaguar, Production, Car" and "Jaguar, Season" which has low overlap and low coverage.

ICR with its adjustable clustering scheme tends to satisfy both the desired criteria of maximal coverage and minimal overlap by identifying hidden classifications like "Jaguar apple system" which is not obviously found from the naturally formed cluster. In fact "Jaguar apple system" makes the query set more meaningful, comprehensive and diverse.

Case Study Example 5 – For Query "Mouse", In general all approaches generate meaningful queries. ICR surprisingly misses out one of the classifications "Mouse, Mickey". This is mainly because, since ICR is also partially based on clusters, ICR restricts itself to find only K clusters. For mouse, the true clusters have 5 clusters which are about "computer mouse",

"album", "cartoons", "results describing experiments with mouse & human gene" and "mouse species family". But since ICR restricts itself to find only 4 clusters, it misses to find one of the classifications. In fact, this is a problem with ICR and all other cluster based approaches, and all of them fail to identify at least one classification if the number of clusters is not properly set.

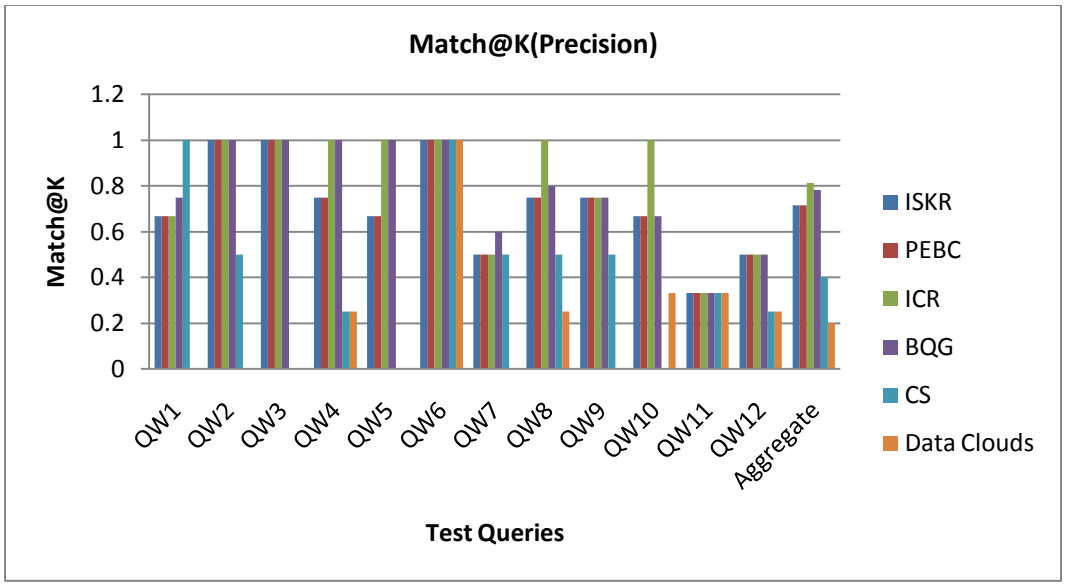
Case Study Example 6 – For Query "Networking", there are basically 3 classifications of documents namely, Camcorders, Camera and Switches. Although the query quality in general is comprehensive and diverse, ISKR and PEBC does not achieve optimal f-measure due to noises in clusters. ICR and BQG overcome the noises and generate queries with perfect f-measure.

Also, note that since TFICF does not considers keyword interaction and cluster quality into account, it generates keywords corresponding to different classification in the same query, for example: One of the expanded queries generated by TFICF is, "Networking, Products:category:routers, Switches:leds:port". Although it reflects that the cluster contains a mix of switches and routers, the user may not find it helpful as it does not aid the user in narrowing down the search.

5.2.4 Match@K

In order to verify the relevance and diversity of topics covered by the expanded queries, this section compares the Match@K [Carmel et al. 2009, Treeratpituk et al., 2006] query values of different systems. Match@K is used by some of the earlier works to compare the cluster labels against a ground

truth of knowledge base. Match@K is defined as the relative number of clusters for which the expanded queries generated are correct. In this framework, an expanded query for a given original query is considered to be correct if it matches any of the labels or description of the labels listed in the Wikipedia's disambiguation page [28] corresponding to the original query. It is more reasonable to consider this as the ground truth because (1) the dataset and test queries used are from Wikipedia, and also (2) Wikipedia's disambiguation pages provide a knowledge repository for different human interpretations of ambiguous queries. The Match@K values are normalized by the number of clusters to maintain the range between [0, 1].



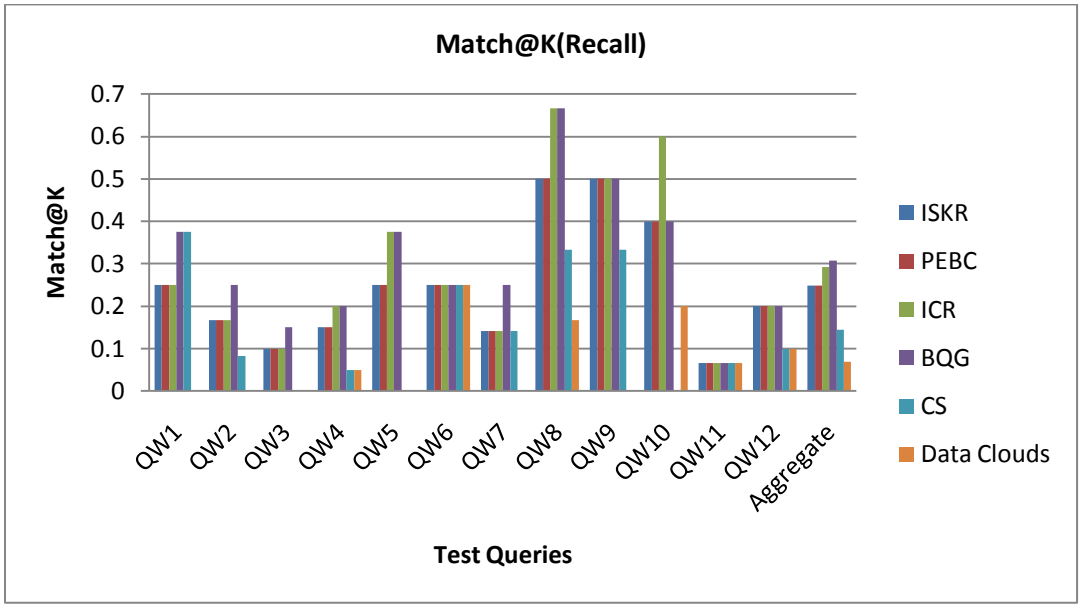


Figure 9 Match@K Values

Note that, for many of the queries, the current work's approaches were able to identify correct labels. For some of the queries such as QW4 "Domino" and QW5 "Eclipse", ISKR and PEBC are not able to generate correct labels for all

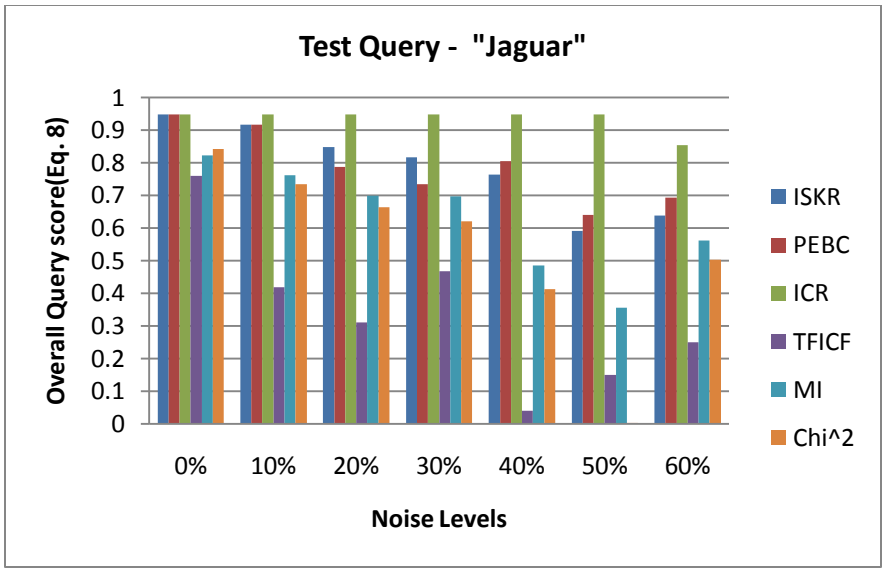
clusters because of imperfect clustering. For such cases, ICR and BQG succeed in identifying correct labels for all clusters. TFICF does not produces consistent results mainly because in some cases it generates too specific cluster labels that may not correspond to broad topic occurring in many documents, for example: For query QW5 "Eclipse", it generates "Eclipse, Java, IBM". Also for some cases, it generates labels with less co-occurrence and thus losing its relevance, for example: For query QW3 "San Jose", it generates "San Jose, team Colorado". Data Clouds most often generates less comprehensive results that manages to cover very few topics.

5.2.5 Noise Resistance

This section discusses the evaluation done to verify the stability of each approach with respect to resistance to noise induced on to the clusters. Noisy clusters are produced by the methods suggested in [Carmel et al. 2009, Treeratpituk et al., 2006]. First, for a given query q, a set of manually classified clusters U is taken which represent the true clusters of the original query results with 0% noise. Then for inducing N% of noise on to the clusters, each result in a cluster C is reassigned to another random cluster in U-C with a probability N(Noise %); with the probability 1-N, the result remains in the same cluster C.

The below figures report the F-measure values (Eq. 8) and Match@K values of three queries, "Jaguar", "Eclipse" and "Canon Products" for different noise levels. The systems compared are ISKR, PEBC, ICR and TFICF as all them involve clustering. ISKR and PEBC tries to generate one expanded query

for each cluster such that the number of results retrieved from the same cluster is maximized and the number of results retrieved from other cluster are minimized. TFICF is a differential cluster labeling algorithm that outputs most frequent term in the cluster in contrast with other clusters as cluster label. ICR tries to minimize the effect of imperfect clusters, by adaptively re-clustering depending on the quality of expanded queries generated. It aims at generating expanded queries that can maximize the coverage of original query results and minimize the overlap of results between expanded queries.



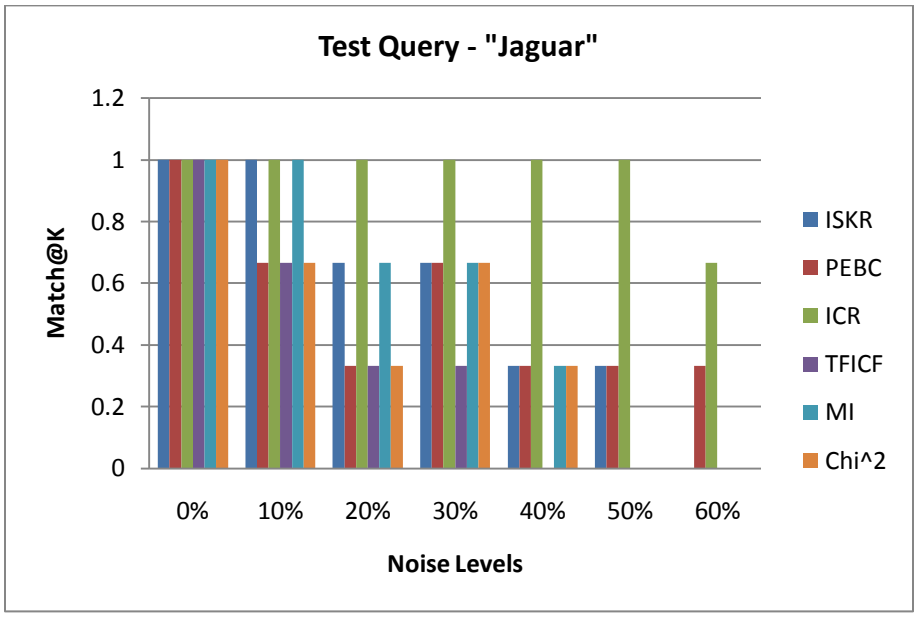
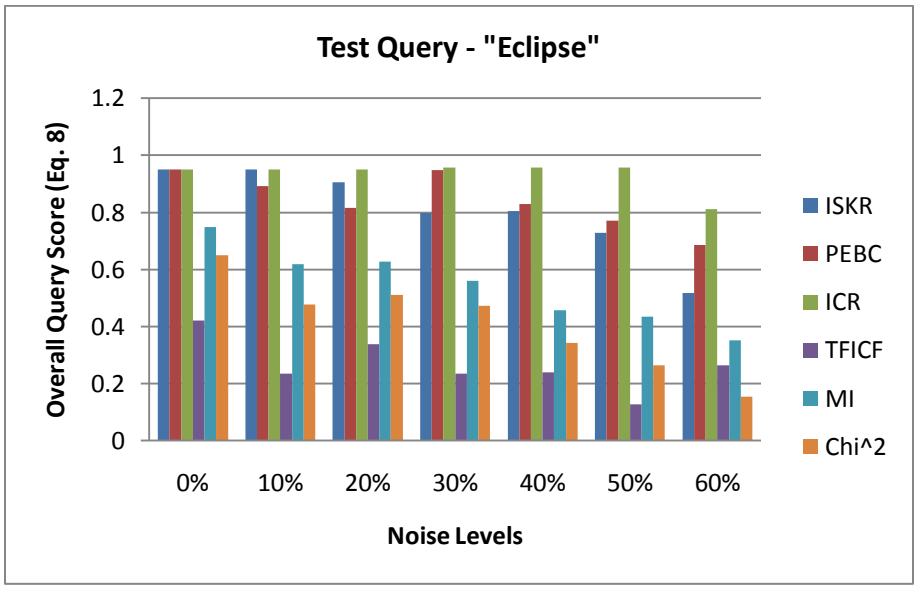


Figure 10 Noise Resistance for Query QW10 "Jaguar"



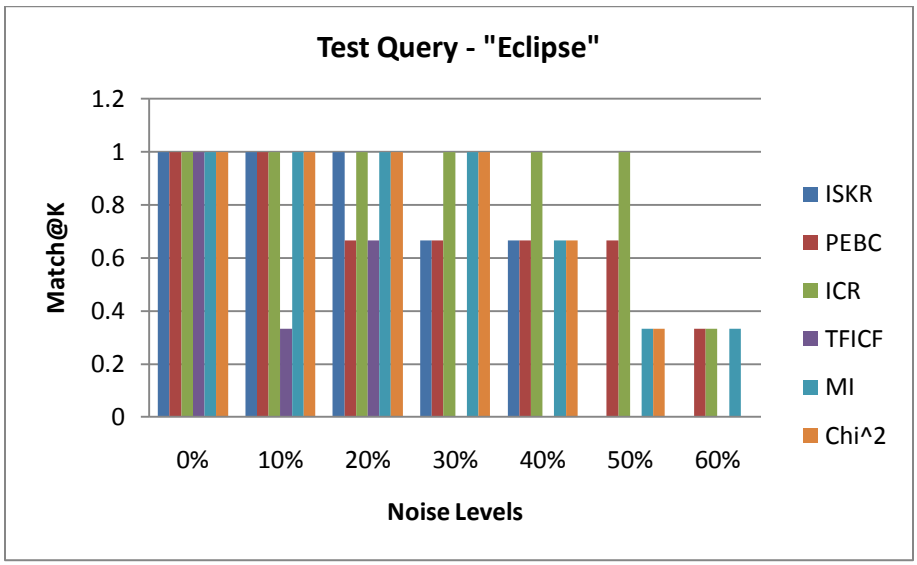
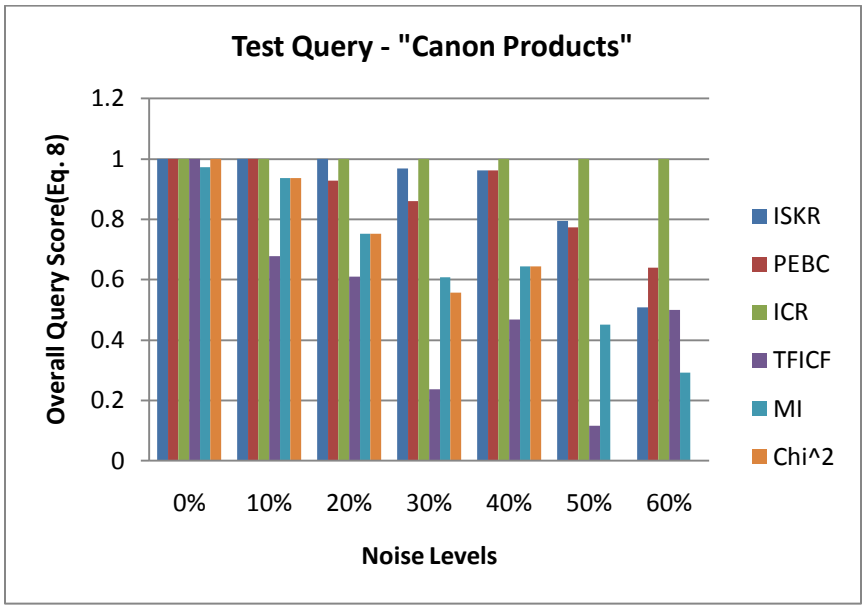


Figure 11 Noise Resistance For Query QW5 "Eclipse"



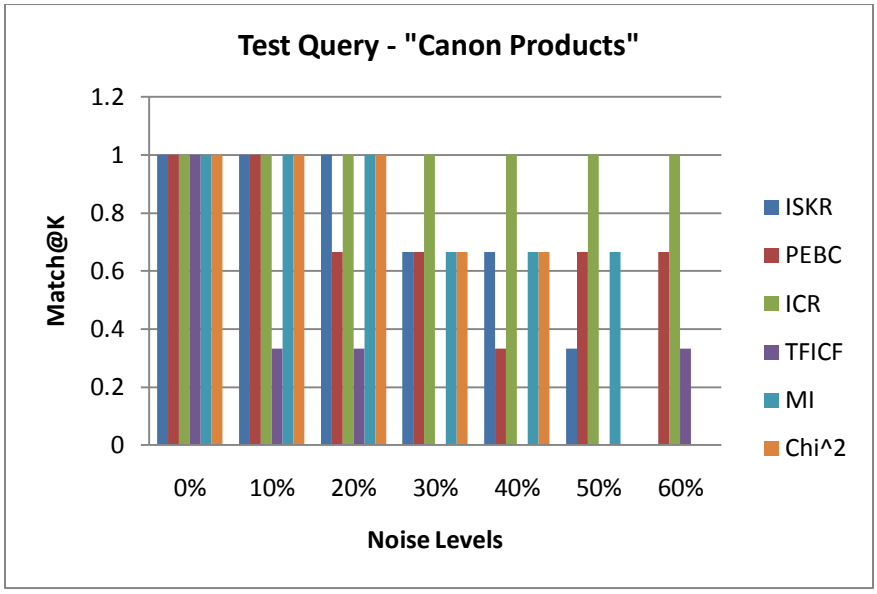


Figure 12 Noise Resistance For Query "Canon Products"

As can be observed from the results above, for all the approaches except ICR, the quality of expanded queries measured by Overall Query Score(Eq. 8) and Match@K values degrade with increasing noise levels. ICR with its adaptive re-clustering scheme tries to maintain the quality of expanded queries and thus achieving better stability even at higher noise levels.

For example: For the query "Jaguar", with noise level 50%, ICR manages to generate expanded queries "Jaguar, Cars", "Jaguar, species", "Jaguar, apple" which correspond to the same three classified clusters at noise level 0%. Thus it shows zero degrade in quality even when the noise levels are as high as 50%. Whereas, ISKR generates queries such as "Jaguar, immediate final", "Jaguar, class" and "Jaguar, automotive UK" at 50% noise level with a quality degrade of 37% in its Overall Query Score, and a quality degrade of 66% in its Match@K score. Similarly, PEBC and TFICF also show quality degrades with increasing noise levels. From the experiments it is observed that, ICR shows greater stability to noise consistently for many queries.

5.3 EFFICIENCY OF QUERY EXPANSION

In the efficiency test, the time taken by each approach is evaluated.

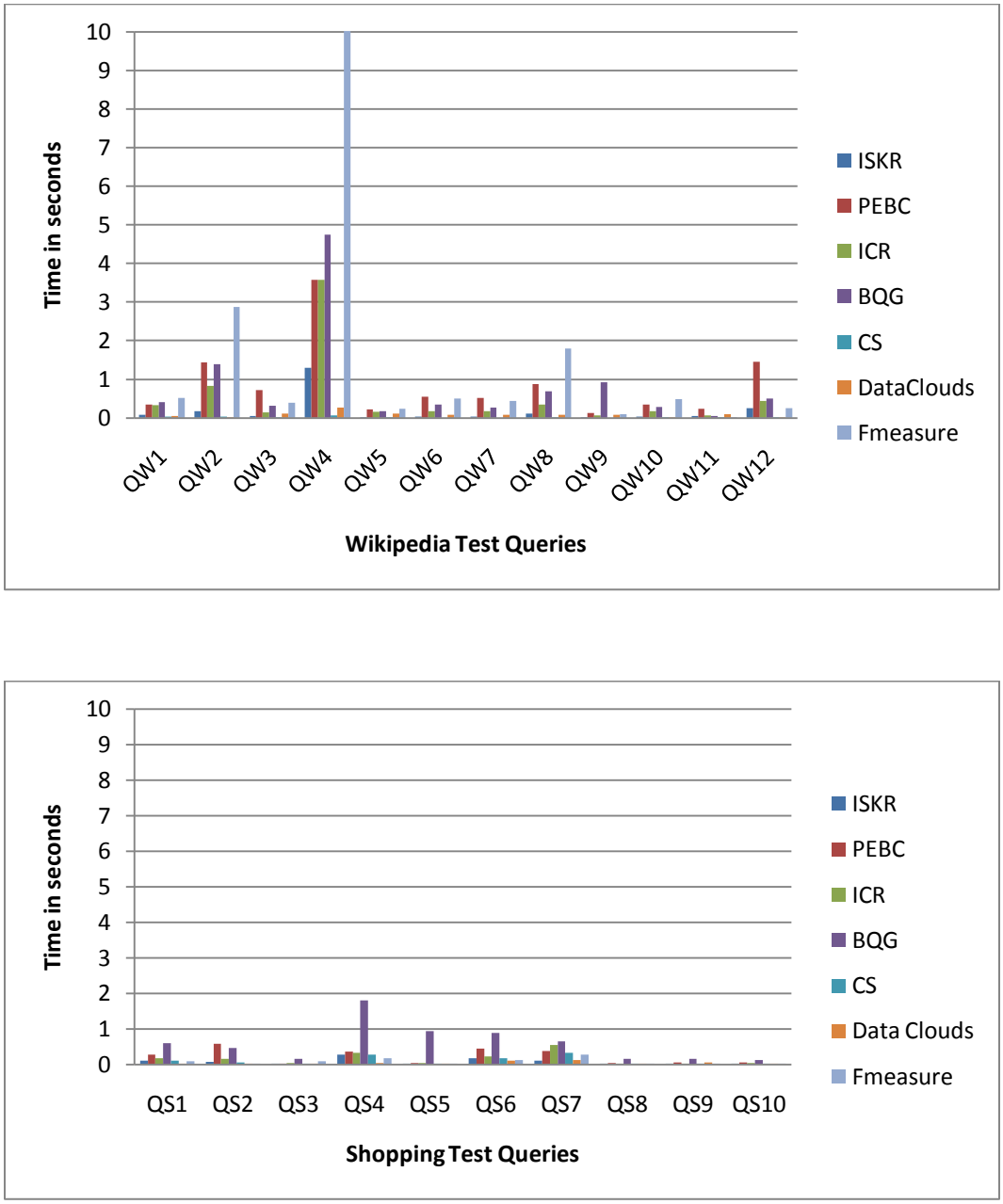


Figure 13: Query expansion time

The time taken includes the clustering time for all approaches except the Data Clouds approach that does not involve any clustering. For each query the number of results selected for processing is 100. For the wiki dataset, each result roughly has about 8000-10000 words, and for the shopping dataset the result size is relatively smaller with about 100 words each. The time taken for the wiki dataset is usually higher than shopping, and ICR takes more time due to the iterative re-clustering that happens in it. Recall that BQG takes more time due to its K^2 complexity, where K is the number of term dimensions. Dimensionality reduction techniques as mentioned in Section 5 have been used to improve the efficiency, and the current implementation scales within 2 seconds for most of the queries for a reasonably larger dataset such as the wiki. As mentioned before, since F-measure approach needs to access every term in the result set and update its value whenever a keyword is added or removed from the query, it takes longer time especially when the size of the result set is large. ISKR prunes this search space and achieves better efficiency by considering only a subset of the terms to update.

Chapter 6 RELATED WORK

Query expansion has been widely studied in literature; the main classes of methods are discussed below.

Query expansion: Expanded queries can be generated based on query log [BarYossef and Gurevich 2008; Chirita et al. 2007], general or domain-specific ontology [Baziz et al. 2005; Grootjen and van der Weide 2006; Fu et al. 2005], user profile and collaboratively filtering [Fu et al. 2005]. Since such

information may not always be available, there are also works that generate expanded queries only based on the information contained in the corpus, i.e., the results retrieved by the user query and/or the entire data repository. As the current work falls into this category, the focus is mainly on discussing corpus-driven query expansion.

There are works that generate new queries based on popular words in the original query result [Cutting et al. 1992, Xu and Croft 1996; Carpineto et al. 2001; Cao et al. 2008; Tao and Yu 2009; Koutrika et al. 2009; Sarkas et al. 2009], considering factors like term frequency, inverse document frequency, ranking of the results in which they appear, etc. In particular, [Koutrika et al. 2009; Tao and Yu 2009] exploit relational databases instead of text documents. [Vechtomova et al. 2003] additionally considers the proximity to the original query keywords when selecting words from results or corpus to compose new queries. As discussed in Section 1 and shown in Section 5, these approaches emphasize on result summarization, and are not suitable for handling exploratory and ambiguous queries.

Relevance Feedback: In relevance feedback, the expanded/refined query aims to retrieve a set of results that are similar to the relevant results, where the relevant results are specified by the user in explicit feedback or are considered to be the top ranked results in pseudo relevance feedback. To generate new queries, various approaches have been proposed to select and rank terms from relevant results, including TFIDF based methods [Koutrika et al. 2009; Xu and Croft 1996], probabilistic language model based methods

[Robertson 1990], vector space model based methods [Xu et al. 2009], etc. However, since users typically provide feedback to top ranked results only, top ranked results are most likely reinforced and the diversity of the results are compromised. Furthermore, the pseudo feedback approach assumes that relevant documents are similar to each other, and are quite different from irrelevant ones. Moreover, relevance feedback approaches do not aim at displaying refined queries to the users and help the users in decision making. These approaches mainly concentrates on refining the results based on the inference obtained from feedback. As a result relevance feedback approaches associate weights with the new queries to fetch appropriate results which cannot be used as expanded queries as these weights are not easily explicable.

Faceted Search: Faceted search provides a classification of the data and enables effective data navigation. There are several approaches for automatically constructing faceted navigation interfaces given the set of query results, which aim at reducing the user's expected navigational cost in finding the relevant results [Chakrabarti et al. 2004; Kashyap et al. 2010; Li et al. 2010]. The current work mainly has the advantage of generating topics from unstructured text documents where it would be difficult for faceted search approaches.

Cluster Labeling / Summarization: The goal of cluster labeling is to find a set of descriptive words for each cluster, which summarizes the content of the cluster, and meanwhile differentiates it from other clusters. Some representative works include [Carmel et al. 2009; Muhr et al. 2010]. A typical

way of measuring the desirableness of a term is TFICF, i.e., term frequency and inverse cluster frequency. Finding cluster representatives for structured data has also been studied. [Jagadish 2009] assumes each result to be a tuple in a relational database with numerical attributes, and uses the k-medoids method to generate a representative for each cluster. Unlike cluster labeling, the interaction of the terms needs to be considered in query expansion, making it a lot more challenging. Furthermore, while cluster labeling quality is typically judged empirically, in this work I propose a quantitative measure of query expansion (i.e. the harmonic mean of the F-measures of the expanded queries).

Relevance feedback can be related to finding expanded queries for each cluster, by considering the current cluster as a set of relevant documents and documents from other clusters as irrelevant documents. Based on this intuition, the original query is refined with more weighted terms that can ideally retrieve only documents corresponding to the relevant set. Some approaches adopt relevance feedback based measures to extract terms.

[Cataldi et al. 2009] groups a set of documents described by "concepts" derived from a known domain taxonomy. Then for each concept, a set of keywords are extracted by considering all documents associated with a concept as relevant and other documents as irrelevant, and then applying probabilistic relevance feedback approach to find terms. This could be related to the current work by considering each concept as a cluster of results and then the problem is to extract terms to represent each cluster.

However, there are several challenges in directly applying this strategy to find representative terms in the current problem setting. Since in the earlier approach, the documents are clustered based on a known taxonomy of concepts, such semantic clustering may be more effective when compared to clustering purely based on distance measures between results as in the current work. As a result, it is more challenging task to find meaningful terms to represent clusters that may not be clearly associated to a concept. Moreover, for describing a concept the earlier approach extracted terms independently without considering the effect of adding other terms on an optimization goal. Our experiments shows that, such approaches generally lead to low recall as two terms may independently dominate a cluster, but together as an expanded query, these terms may not retrieve many results. Semantically also, such expanded queries are less meaningful as they may correspond to multiple categories (For example: "Cell, Cell wall, Cell table") and the users will find it difficult to narrow down the search to a desired result. The current approach decides to add/remove more terms from an expanded query with the aim of optimizing the goal function further, thus following a more systematically guided approach.

[Kim et al. 2009] proposed methods to automatically tag blogs, such that the tags are general and shared by other related blogs as well as discriminating in order preserve the differences between the blogs. This philosophy can be related to the current work's goal to find expanded queries that are general in retrieving more results from the cluster as well as specific to ensure not retrieving results from other clusters.

Advances in the works on automatic image annotations/tagging also have some relation to the current work. For example, the approach of block based image tagging model [Mori et al. 1999, Jeon et al. 2003], extract feature vectors from a given image and cluster these features to separate out the visual objects found in the image. These visual objects are then mapped to representative set of visual words by making use of an Image object keyword library. Recent developments in this field improve the quality of annotations by considering the correlation between keywords. The Progressive image annotation model [Wang et al. 2007], pick the best word to add to the annotation at each stage based on the joint probability of words already in the annotation, which can lead to the greatest increase in the objective function.

Compared with existing work, there is several uniqueness of the current work. First, compared with existing query expansion approaches, in this work expanded queries are generated with the aim of presenting a classification of the original query results. This is especially useful for handling exploratory queries and ambiguous queries. Second, the technical contributions focus on how to generate queries with high F-measure given the ground truth of query result. To the best of my knowledge, this is the first study on this problem. Furthermore, unlike existing work that addresses the query expansion problem using heuristics; this work formalizes the problem and quantifies the quality of an approach.

REFERENCES

[1] Amazon Mechanical Turk: https://www.mturk.com/mturk/welcome

[2] B. L. and Jagadish, H. V. 2009. Using Trees To Depict A Forest. PVLDB 2, 1, 133–144.

[3] Bar-Yossef, Z. and Gurevich, M. 2008. Mining Search Engine Query Logs Via Suggestion Sampling. PVLDB 1, 1, 54–65.

[4] Baziz, M., Boughanem, M., and Aussenac-Gilles, N. 2005. Conceptual Indexing Based On Document Content Representation. In CoLIS. 171–186.

[5] Broder, A. Z. 2002. A Taxonomy of Web Search. SIGIR Forum 36, 2, 3–10. Cao, G., Nie, J.-Y., Gao, J., and Robertson, S. 2008. Selecting Good Expansion Terms For Pseudo-Relevance Feedback. In SIGIR. 243–250.

[6] Carmel, D., Roitman, H., and Zwerdling, N. 2009. Enhancing Cluster Labeling Using Wikipedia. In SIGIR. 139–146.

[7] Carpineto, C., de Mori, R., Romano, G., and Bigi, B. 2001. An Information-Theoretic Approach to Automatic Query Expansion. ACM Trans. Inf. Syst. 19, 1, 1–27.

[8] Cataldi, M., Schifanella, C., Candan, K.S., Sapino, M.L., Caro, L.D., CoSeNa: a context-based search and navigation system. MEDES 2009: 218-225

[9] Chakrabarti, K., Chaudhuri, S., and won Hwang, S. 2004. Automatic Categorization of Query Results. In SIGMOD Conference. 755–766.

[10] Chirita, P.A., Firan, C. S., and Nejdl, W. 2007. Personalized Query Expansion for the Web. In SIGIR. 7–14.

[11] Cutting, D.R., Karger, J.O. Pedersen, and J.W.Tukey. Scatter/gather: a cluster-based approach to browsing large document collections. In SIGIR '92, pages. 318–329, New York, NY, USA, 1992. ACM

[12] Faloutsos, C., and Link, K.I. FastMap: A Fast Algorithm for Indexing, Data-Mining and Visualization of Traditional and Multimedia Datasets. In SIGMOD '95, San Jose, CA, USA, 1995. ACM

[13] Fu, G., Jones, C. B., and Abdelmoty, A. I. 2005. Ontology-Based Spatial Query Expansion in Information Retrieval. In OTM Conferences (2). 1466–1482.

[14] Fu, L., Goh, D. H.-L., and Foo, S. S.-B. 2005. Evaluating the Effectiveness of a Collaborative Querying Environment. In ICADL. 342–351.

[15] Geraci, F., Pellegrini, M., Maggini, M., Sebastiani, F. Cluster Generation and Cluster Labelling for Web Snippets: a Fast and Accurate Hierarchical Solution, Internet Mathematics, 2007.

[16] Grootjen, F. A. and van der Weide, T. P. 2006. Conceptual Query Expansion. Data Knowl. Eng. 56, 2, 174–193.

[17] Huang, Y., Liu, Z., and Chen, Y. 2008. Query Biased Snippet Generation in XML Search. In SIGMOD Conference. 315–326.

[18] Jeon J., Lavrenko V., and Manmatha R., Automatic Image Annotation and Retrieval using Cross-Media Relevance Models In Proceedings of the 26th Intl. ACM SIGIR Conf., pages 119–126, 2003

[19] Kashyap, A., Hristidis, V., and Petropoulos, M. 2010. FACeTOR: Cost-Driven Exploration of Faceted Query Results. In CIKM. 719–728.

[20] Kim, J.K., Candan, K.S., Tatemura, J. Organization and Tagging of Blog and News Entries Based on Content Reuse. Signal Processing Systems 58(3): 407-421 (2010)

[21] Koutrika, G., Zadeh, Z. M., and Garcia-Molina, H. 2009. Data Clouds: Summarizing Keyword Search Results over Structured Data. In EDBT. 391–402.

[22] Li, C., Yan, N., Roy, S. B., Lisham, L., and Das, G. 2010. Facetedpedia: Dynamic generation of query-dependent faceted interfaces for wikipedia. In WWW. 651–660.

[23] Liu, Z., Sun, P., and Chen, Y. 2009. Structured Search Result Differentiation. PVLDB 2, 1, 313–324.

[24] Liu T., Liu S., Chen Z., Ma W.Y., An Evaluation on Feature Selection for Text Clustering, ICML 2003, Washington DC, 2003.

[25] C. D. Manning, P. Raghavan, and H. Schutze. Introduction to Information Retrieval. Cambridge University Press, 2008.

[26] Mori Y., Takahashi H., and Oka R., Image-to-word transformation based on dividing and vector quantizing images with words. In MISRM'99: First International Workshop on Multimedia Intelligent Storage and Retrieval Management, 1999.

[27] Muhr, M., Kern, R., and Granitzer, M. 2010. Analysis of Structural Relationships for Hierarchical Cluster Labeling. In SIGIR. 178–185.

[28] D. R. Radev, H. Jing, M. Sty´s, and D. Tam. Centroid-based summarization of multiple documents. Information Processing Management, 40(6):919–938, 2004.

[29] Robertson, S. E. 1990. On Term Selection for Query Expansion. Journal of Documentation 46, 359–364.

[30] Sarkas, N., Bansal, N., Das, G., and Koudas, N. 2009. Measure-driven keyword-query expansion. PVLDB 2, 1, 121–132.

[31] Tao, Y. and Yu, J. X. 2009. Finding Frequent Co-occurring Terms in Relational Keyword Search. In EDBT. 839–850.

[32] W. S. Torgerson. Multidimensional scaling: I. theory and method. Psychometrika, 17:401–419, 1952.

[33] Treeratpituk, P., Callan, J. Automatically Labeling Hierarchical Clusters. In DG.O '06, pages 167–176, NewYork, NY, USA, 2006. ACM.

[34] Tseng Y.H, Lin C.J, Chen H.H, Lin Y.I. Toward Generic Title Generation for Clustered Documents, AIRS 2006.

[35] http://en.wikipedia.org/wiki/Wikipedia:Links_to_(disambiguation)_pages

[36] Wang B., Li Z., Yu N., Li M., Image annotation in a progressive way. In In proc. ICME, pages 811–814, 2007.

[37] Vechtomova, O., Robertson, S. E., and Jones, S. 2003. Query Expansion with Long-Span Collocates. Inf. Retr. 6, 2, 251–273.

[38] Xu, J. and Croft, W. B. 1996. Query Expansion Using Local and Global Document Analysis In SIGIR. 4–11.

[39] Xu, Y., Jones, G. J. F., and Wang, B. 2009. Query Dependent Pseudo-Relevance Feedback based on Wikipedia. In SIGIR. 59–66

[40] Yang Y., Pederson O.J, A Comparative Study on Feature Selection in Text Categorization. In ICML 1997, pp. 412-420.

APPENDIX A

TEST QUERIES

Wikipedia	
QW1	CVS
QW2	Columbia
QW3	San Jose
QW4	Domino
QW5	Eclipse
QW6	Java
QW7	Cell
QW8	Rockets
QW9	Mouse
QW10	Jaguar
QW11	Greek
QW12	Drama

Shopping	
QS1	Canon Products
QS2	Networking Products
QS3	Routers
QS4	TV
QS5	TV Plasma
QS6	HP Products
QS7	Memory
QS8	Memory 8GB
QS9	Memory Internal
QS10	Printer

APENDIX B

PSUEDOCODE

Algorithm 1 – Iterative Single keyword Refinement

$ISKR\ (User\ Query: uq, Cluster: C, Results\ not\ in\ C: U)$
1: $K = the\ set\ of\ keywords\ in\ C \cup U$
2: $q = uq$
3: $Refine(C;\ U;\ K;\ q;\ weight)$
4: $return\ q$

$Refine\ (C;\ U;\ K;\ q;\ weight))$
1: $T = \emptyset$
2: $for\ each\ k \in K, k =\in q\ do$
    3: $E(k) = the\ set\ of\ results\ that\ do\ not\ contain\ k$
    4: $benefit(k) = S\big(R(q) \cap C \cap E(k)\big)$
    5: $cost(k) = S(R(q) \cap U \cap E(k))$
    6: $value(k) = benefit(k) = cost(k)$
    7: $insert\ k\ into\ T$
8: $for\ each\ k \in K, k \in q\ do$
    9: $D(k) = R(q\backslash k)\backslash R(q)$
    10: $benefit(k) = S(D(k) \cap C)$
    11: $cost(k) = S(D(k) \cap U)$
12: $while\ true\ do$
    13: $k = top - 1\ keyword\ in\ T$
    14: $if\ value(k) \leq 1\ then$
        15: $break$
    16: $if\ k \in q\ then$
        17: $q = q\backslash k$
        18: $MaintainT\ (T;\ q;\ k;\ E(k);\ K;\ C, remove)$
    19: $else$
        20: $q = q \cup k$
        21: $MaintainT\ (T;\ q;\ k;\ E(k);\ K;\ C, add)$
22: $return\ q$

$MaintainT\ (T;\ q;\ k;\ E(k);\ K;\ C;\ type)$
1: $if\ type = add\ then$
    2: $deltaResult = R(q\backslash k) \cap E(k)$
3: $else$
    4: $deltaResult = R(q\backslash k)\backslash R(q)$
5: $for\ each\ k' \in K\ do$
    6: $if\ each\ k'appears\ in\ all\ results\ in\ deltaResult\ then$
    7: $continue$
8: $if\ type = add\ then$
    9: $benefit\big(k'\big) = R(q) \cap U \cap E\big(k'\big)$
    10: $cost(k') = R(q) \cap C \cap E(k')$
11: $else$
    12: $D(k) = R(q\backslash k)\backslash R(q)$
    13: $benefit(k') = D(k) \cap C$

14: $cost(k') = D(k) \cap U$
15: $remove\ k'\ from\ T$
16: $value(k') = benefit(k') = cost(k')$
17: $add\ k'\ to\ T$

Algorithm 2 – Partial Elimination Based Convergence

$PEBC\ (User\ Query: uq, Cluster: C, Results\ not\ in\ C: U)$
1: $K = the\ set\ of\ keywords\ in\ C \cup U$
2: $q = uq$
3: $Converge(C;\ U;\ K;\ q)$
4: $return\ q$

$Converge\ (C;\ U;\ K;\ q))$
1: $nseg = 5$ {$set\ the\ number\ of\ segments\ to\ split\ the\ interval$}
2: $nit = 5$ {$set\ the\ number\ of\ iterations$}
3: $left = 0, right = 100, step = (right - left) = nseg$
4: $for\ i = 1\ to\ nit\ do$
5: $for\ x = left;\ x \leq right;\ x+= step\ do$
6: $currC = C, currU = U$
7: $repeat$
8: $r = a\ randomly\ selected\ result$
9: $bestvalue = 0$
10: $for\ each\ distinct\ keyword\ k \in r\ do$
11: $E(k) = the\ set\ of\ results\ that\ do\ not\ contain\ k$
12: $benefit(k) = E(k) \cap U$
13: $cost(k) = E(k) \cap C$
14: $value(k) = benefit(k) = cost(k)$
15: $if\ value(k) > bestvalue\ then$
16: $selecetd = k;\ bestvalue = value(k)$
17: $q = q \cup selected$
18: $currC = C \backslash E(k), currU = U \backslash E(k)$
19: $until\ roughly\ x\%\ percent\ of\ results\ in\ U\ are\ eliminated$
20: $left;\ right = the\ interval\ with\ the\ largest\ average\ score$

Algorithm 3 – Iterative Cluster Refinement

$ICR\ (User\ Query: uq, Results: R, Number\ of\ Expanded\ Queries: k)$
1: $finalizedQueries = \emptyset$
2: $while\ finalizedQueries: size < k\ do$
3: $numOfCluster = k - finalizedQueries: size$
4: $candidateQueries =$
$QEC(uq;\ R - R(finalizedQueries);\ numOfCluster;\ R(finalizedQueries))$

{$generate\ expanded\ queries\ using\ the\ algorithm\ for\ QEC\ problem\ (ISKR\ or\ PEBC)$}

5: $let\ q\ be\ a\ query\ in\ candidateQueries\ with\ the\ best\ desirableness\ according\ to\ Eq.6$
6: $finalizedQueries = finalizedQueries \cup q$

Algorithm 4 – Bisecting Query Generation

$BCG\ (User\ Query: uq, Results: R, Lower\ Bound: lb, Upper\ Bound: ub, Threshold\ Th)$
1: $K\ =\ the\ set\ of\ keywords\ in\ R$
2: $for\ i\ =\ 1\ to\ |K|\ do$
    3: $for\ j\ =\ i\ +\ 1\ to\ |K|\ do$
        $s(ki\ , kj\ )\ =\ score(\{ki\ \}, \{kj\ \}, R)\ according\ to\ Eq.8$

5: $let\ ki\ , kj\ be\ queries\ in\ K\ such\ that\ s(ki\ , kj\ )\ is\ maximized$
6: $expandedQueries\ =\ \{ki\ \}, \{kj\ \}$
7: $while\ expandedQueries.size\ <\ ub$
  8: $If\ km\ \epsilon\ K\ can\ \ improve\ the\ score\ s(\{expandedQueries\}, \{km\})\ atleast\ by\ Th\ , then:$
    9: $expandedQueries = expandedQueries\ \cup km$
  10: $else\ if\ expandedQueries.size < lb$
    11: $Let\ q\ be\ the\ query\ in\ expandedQueries\ with\ highest\ desirableness\ according\ to\ Eq.16$
    12: $for\ i\ =\ 1\ to\ |K - q|\ do$
      13: $for\ j\ =\ i\ +\ 1\ to\ |K - q|\ do$
        14: $s(ki, kj) =\ score\big(q \cup \{ki\}, q \cup \{kj\}, R(q)\big) according\ to\ Eq.8$
        15: $Let\ ki, kj\ be\ the\ queries\ in\ K - q, such\ that\ s(ki, kj) is\ maximized.$
        16: $expandedQueries = expandedQueries - q$
        17: $expandedQueries = expandedQueries\ \cup (q \cup ki) \cup (q \cup kj)$
  18: $else: break;$

APPENDIX C

APX-HARDNESS OF QEC PROBLEM

In this section, the QEC problem is shown to be as hard as the Independent Set problem in terms of approximation. Recall that, the Independent Set problem is to find the maximum Set of nodes in an undirected graph, such that no two nodes are connected by an edge. The Independent Set problem has been proved to be APX-Hard, i.e., it has no constant approximation ratio. It will be shown in this section that if the QEC problem has an approximation ratio of K, then the Independent Set problem has an approximation Ratio of 4k-3.

Given any instance of the independent set problem: an undirected graph, $G(V,E), let\ n = V, m = E$, create an instance of the QEC problem as follows. Assume that each node in G has at least one edge (otherwise it can be directly added to the independent set).

1. Create n+1 keywords, $K0, \dots, Kn$
2. Create two clusters C1 and C2. C1 has n results, and C2 has $4Kn^2m$ results. In C2, let the first $4Kn^2$ results be in group 1, the second $4Kn^2$ results be in group 2 and the $mth\ 4Kn^2$ be in group m.
3. Let keyword K0 appear in all results in C2 but none of the results in C1.
4. Let keyword $Ki(1 <= i <= n)$ appear in all results in C1 except the ith result.
5. For each node i, if it has x edges, then let keyword ki not appear in the corresponding x blocks of results in C2. For example, if node i has edges e1, e3, e6, then ki does not appear in results in groups 1, 3, 6, in C2, and appears in all other results in C2.

This is an instance of QEC. It is easy to see that the optimal query for C2 is $q2 = \{k0\}$, which means $F - measure(q2) = 1$. Let q1 be the optimal query for C1. First it will be proved that, $precision(q1)$ must be 1, i.e., q1 cannot retrieve any results in C2. In this case, q1 is a query such that it eliminates all results in C2 using the least number of keywords (note that the more keywords q1 uses, the less results it retrieves in C1). It is important to notice that q1 must not contain all keywords $ki(1 \le i \le n)$. The reason is that, if we pick any edge $ei(na, nb)\ \epsilon\ G$, results in group i in C2 can be eliminated using either keyword ka or keyword kb, thus either ka or kb does not need to be in q1. Therefore, q1 at least retrieves one result in C1, and $recall(q1) \ge 1/|C1| = 1/n;$

$$S1 = score(q1, q2) \geq \frac{2}{\left(\frac{n+1}{2}\right)+1} = \frac{4}{n+3}$$

On the other hand, if $precision(q1) \neq 1$, q1 must retrieve some results in C2. Since results in C2 has m groups, each group containing $4kn$^2 identical results, q1 must retrieve at Least $4kn$^2 results in C2, thus:

$$precision(q1) \leq \frac{n}{4kn^2} = \frac{1}{4kn}$$

$$F - measure \leq \frac{2}{4kn+1}$$

$$S2 = score(q1, q2) \leq \frac{2}{\frac{(4kn+1)}{2}+1} = \frac{4}{4kn+3}$$

Since

$$S2 \leq \frac{4}{4kn+3} \leq \frac{4}{n+3} \leq S1$$

S2 cannot be the optimal score of q1 and q2, thus $precision(q1)$ must be 1. Next it will be shown that, for any arbitrary k-approximate solution for this QEC instance, consisting of queries q1' and q2', where $2' = \{k0\}$, $precision(q1') = 1$. Note that,

$$S2 \times k \leq \frac{4k}{4kn+3} < \frac{4k}{kn+3k} = \frac{4}{n+3} \leq S1 = OPT$$

Thus unless $precision(q1) = 1$, it cannot approximate the optimal solution within k.

Suppose we have an algorithm that can give a k-approximate solution of the above instance in polynomial time, and then now we will see how to get an approximate solution for the independent set problem with ratio 4k-3. Let R, F and S denote $recall(q1)$, $F - measure(q1)$ and $score(q1, q2)$ in the optimal solution, and R', F' and S' denote the corresponding values in the approximate solution. We have $k \times S' \geq S$. Since $S = \frac{2F}{1+F}$ and $S' = \frac{2F'}{1+F'}$, we have

$$\frac{2kF'}{1+F'} \geq \frac{2F}{1+F}$$

$$kF'(1+F) \geq F(1+F')$$

$$F'(k + kF - F) \geq F$$

$$\frac{F'}{F} \geq \frac{1}{k + (k-1)F} \geq \frac{1}{2k-1}$$

And since $F = \frac{2R}{1+R}$ AND $F = \frac{2R'}{(1+R')}$, we have $k' = 2k - 1$, we have

$$\frac{2k'R'}{1+R'} \geq \frac{2R}{1+R}$$

Using same deduction above, we get

$$\frac{R'}{R} \geq \frac{1}{2k'-1} = \frac{1}{4k-3} \qquad (1)$$

Note that the optimal solution, q1 is the query which eliminates all results in C2 using the minimum number of keywords. Let the number of keywords in q1 be P. Since each keywords in q1 eliminates a result in C1, q1 retrieves $n - P$ results in C1, thus $R = \frac{n-P'}{n}$. According to Eq. 1 we have,

$$\frac{n-P'}{n-P} \geq \frac{1}{4k-3}$$

Now let us look at the Independent Set instance. Recall that each node in G corresponds to a keyword in C1 in the QEC instance. Note that the set of nodes corresponding to the keywords in P comprises the minimal vertex cover of G. Because if an edge $ei(na, nb)$ is not covered (i.e., neither $or\ kb \epsilon P$), then since results in $group\ i\ in\ c2$ only misses keywords $ka\ and\ kb$, these results are retrieved by q1, Which is contradictory with $precision(q1) = 1$. Therefore, the set of nodes corresponding to the keywords not in P comprises the maximal independent set of G, whose size is $n - P$. Similarly, the set of nodes corresponding to the keywords not in p0 comprises an approximate independent set of G, whose size if $n - P$'. According to equation 2, we have obtained a 4k-3 approximate solution for this Independent Set instance. Since this is an arbitrary independent set instance, it contradicts with the conclusion that Independent Set instance is APX-hard. Therefore, the QEC problem is APX-hard.

APPENDIX D

EXPANDED QUERIES

QW1: CVS		**QW2: Columbia**	
ISKR	Query 1: CVS, member Query 2: CVS, store Query 3: CVS, software	ISKR	Query 1: Columbia, new York Query 2: Columbia, British
PEBC	Query 1:CVS, member Query 2:CVS, stores Query 3:CVS software	PEBC	Query 1: Columbia, new York Query 2: Columbia, British
ICR	Query 1:CVS, store Query 2:CVS, software Query 3:CVS, association	ICR	Query 1: Columbia, British Query 2: Columbia, new York
BQG	Query 1: CVS, pharmacy Query 2: CVS , support Query 3: CVS, GNU Query 4: CVS, member	BQG	Query 1: Columbia, British Query 2: Columbia, new York, university Query 3: Columbia, new York, musical
TFICF	Query 1: CVS ships aircraft Query 2: CVS stores convenience Query 3: CVS software Linux	TFICF	Query 0: Columbia Sony actress Query 1: Columbia provincial British
Data Clouds	Query1: CVS, town Query2: CVS, blue Query3: CVS, fire	Data Clouds	Query 0: Columbia CBS Query 1: Columbia blue
Google	Query1: CVS, files Query2: CVS, client Query3: CVS, Wikipedia	Google	Query 1: Columbia, country Query 2: Columbia, facts Query 3: Columbia, pictures
Measure	Query 1: CVS member Query 2: CVS store pharmacy Query 3: CVS software	F measure	Query 1: Columbia new York Query 2: Columbia British

QW3: San Jose		**QW4: Domino**	
ISKR	Query 1: San José, California Query 2: San José, province	ISKR	Query 0:records album Query 1:company Query 2:products Query 3:database
PEBC	Query 1:Sanjose, California Query 2:Sanjose, province	PEBC	Query 0:domino, album records Query 1:domino, companies Query 2:domino, border Query 3:domino, database
ICR	Query 1: San José, Santa Query 2: San José, team , players	ICR	Query 1:domino, products Query 2:domino, album Query 3:domino, database notes lotus Query 4:domino, pizza UK
BQG	Query 1: sanjose,team Query 2: sanjose,santa, players Query 3: sanjose, santa, california	BQG	Query 0: Domino, album Query 1: Domino, database Query 2: Domino, products, Lexus Query 3: Domino, products, International
TFICF	Query 0: sanjose bruno avenue Query 1: sanjose colorado team	TFICF	Query 1: domino, album squeeze Query 2: domino, investment pizza Query 3: domino, Lexus border Query 4: domino lotus notes
Data Clouds	Query 0: san José, military Query 1: san José, anti	Data Clouds	Query 1: domino, playing Query 2: domino, blue Query 3: domino, operations Query 4: domino, audio
Google	Query 1: san José, attractions Query 2: san José, airport Query 3: san José, sharks	Google	q1: Domino, game q2: Domino, movie q3: Domino, rapper
F measure	Query 1: san José, California Query 2: san José, province	F measure	Query 0: domino playing Query 1: domino blue Query 2: domino operations Query 3: domino audio

QW5: Eclipse		**QW6: Java**	
ISKR	Query 1:Eclipse, software Query 2: Eclipse, night Query 3: Eclipse, solar	ISKR	Query 1: Java, Indonesia Query 2: Java, software
PEBC	Query 1: Eclipse, software Query 2: Eclipse, night Query 3: Eclipse, solar	PEBC	Query 1: Java, Indonesia Query 2: Java, software
ICR	Query 1: Eclipse, software Query 2: Eclipse, solar Query 3: Eclipse, car	ICR	Query 1: Java, Indonesia Query 2: Java, software
BQG	Query 1: Eclipse, solar Query 2: Eclipse, software Query 3: Eclipse, car Query 4: Eclipse, albums Query 5: Eclipse, derby	BQG	Query 1: Java, software Query 2: Java, Indonesia Query 3: Java, implementation
TFICF	Query 1: Eclipse, java IBM Query 2: Eclipse, horror concert Query 3: Eclipse, Mathematics astronomical	TFICF	Query 1: Java, Indonesia Query 2: Java, software
Data Clouds	Query 1: Eclipse, points Query 2: Eclipse, Indian Query 3: Eclipse, ratio	Data Clouds	Query 1: Java, Towns Query 2: Java, JavaScript
Google	Query 1: Eclipse, car Query 2: Eclipse, book Query 3: Eclipse, fan subs	Google	Query 1: Java, quote Query 2: Java, script

F measur e	Query 1: Eclipse, software Query 2 Eclipse, night Query : Eclipse, solar	F measure	Query 0: Java, Indonesia Query 1: Java, software

QW7: Cell		**QW8: Rockets**	
ISKR	Query 1:Cell, human gene Query 2:Cell, biology Query 3: Cell, anode Query 4: Cell, wall	ISKR	Query 1:launch Query 2:league Query 3:missiles Query 4:band
PEBC	Query 1:Cell, proteins Query 2:cell, system Query 3:cell, anode Query 4:cell, wall	PEBC	Query 1:rockets, weapon Query 2:rockets, players Query 3:rockets, fuel Query 4:rockets, band
ICR	Query 1:cell, wall Query 2:cell, anode cathode Query 3:cell, tumor Query 4:cell, human	ICR	Query 1:rockets, missiles Query 2:rockets, album Query 3:rockets, league Query 4:rockets, weapon German
BQG	Query 1: cell, system Query 2: cell, molecular Query 3: cell, adenocarcinoma Query 4: cell, solution Query 5: cell, header	BQG	Query 1: rockets, league Query 2: rockets, launch Query 3: rockets, band Query 4: rockets, fire Query 5: rockets, pc games
TFICF	Query 1: cell carcinoma entrez Query 2: cell mantle leukemia Query 3: cell cathode anode Query 4: cell row table	TFICF	Query 1: rockets games artillery Query 2: rockets coach basketball Query 3: rockets space system Query 4: rockets Israeli launch
Data Clouds	Query1: Cell, marrow Query2: Cell, cluster Query3: Cell, acid Query4: Cell, primary	Data Clouds	Query 1: rockets olajuan Query 2: rockets German Query 3: rockets launch Query 4: rockets space

Google	Query1: Cell, parts of a cell Query2: animal cell Query3: plant cell Query4: cell dbz	Google	Query 1: model rockets Query 2: toy rockets Query 3: rockets rockettes Query 4: rockets pictures
F measure	Query 0: cell sites Query 1: cell biology Query 2: cell system Query 3: cell wall	F measure	Query 1: rockets launch Query 2: rockets league Query 3: rockets missiles Query 4: rockets band

QW9: Mouse		**QW10: Jaguar**	
ISKR	Query 1:Mouse, family Query 2:Mouse, Disney Query 3:Computer mouse Query 4:Mouse, album	ISKR	Query 1: Jaguar tiger Query 2: Jaguar season Query 3: Jaguar production car
PEBC	Query 1:mouse, family Query 2:mouse, Disney Query 3:mouse, computer Query 4:mouse, album	PEBC	Query 1:jaguar, tiger Query 2:jaguar, season Query 3:jaguar, car
ICR	Query 1:mouse, computer Query 2:mouse, album Query 3:mouse, gene human Query 4:mouse, species family	ICR	Query 1:jaguar, production car Query 2:jaguar, apple system Query 3:jaguar, species
BQG	Query 1: mouse, computer Query 2: mouse, family Query 3: mouse, Mickey Query 4: mouse, singles	BQG	Query 1: jaguar, engine Query 2: jaguar, species Query 3: jaguar, studio
TFICF	Query 1: mouse human species Query 2: mouse Mickey Disney Query 3: mouse input computer Query 4: mouse modest guitar	TFICF	Query 1: jaguar os nova Query 2: jaguar flag ret Query 3: jaguar lexus mg
Data Clouds	Query 1: mouse , cdna Query 2: mouse , blue Query 3: mouse , multimammate Query 4: mouse , domain	Data Clouds	Query 1: jaguar MAC Query 2: jaguar industry Query 3: jaguar design

Google	Query 1: mouse , house Query 2: pictures of mice Query 3: Logitech mouse Query 4: mouse pictures	Google	Query 1: jaguar models Query 2: jaguar car Query 3: jaguar animal
F measure	Query 1: mouse family Query 2: mouse Disney Query 3: mouse computer Query 4: mouse album	F measure	Query 1: jaguar tiger Query 2: jaguar season Query 3: jaguar production

QW11: Greek		**QW12: Drama**	
ISKR	Query 0: Greek, ancient Query 1: Greek, mythology Query 2: Greek, Greece	ISKR	Query 0: drama, television Query 1: drama, love Query 2: drama, film Query 3: drama, arts
PEBC	Query 0: Greek, ancient Query 1 Greek, mythology Query 2: Greek, Greece	PEBC	Query 0:drama, television Query 1:drama, play Query 2:drama, film Query 3:drama, short
ICR	Query 1: Greek, mythology Query 2: Greek, Greece Query 3: Greek, ancient	ICR	Query 1:drama, film Query 2:drama, radio Query 3:drama, plays Query 4:drama, television cast
BQG	Query 0: Greek, ancient Query 1: Greek, Greece Query 2: Greek, mythology	BQG	Query 0: drama, television Query 1: drama, plays Query 2: drama, director, movie Query 3: drama, director, theatre
TFICF	Query 0: Greek, church ancient Query 1: Greek, mythology ancient Query 2: Greek, team league	TFICF	Query 0: drama achievement role Query 1: drama records cd Query 2: drama horror comedy Query 3: drama poetry tvb
Data Clouds	Query 0: Greek mythology Query 1: Greek museum Query 2: Greek company	Data Clouds	Query 0: drama artists Query 1: drama writing Query 2: drama cuba Query 3: drama perform

Google	Query 0: Greek show Query 1: Greek wiki Query 2: Greek translation	Google	Query 0: drama plays Query 1: types of drama Query 2: korean drama Query 3: drama definition
F measure	Query 0: Greek ancient Query 1: Greek mythology Query 2: Greek Greece	F measure	Query 0: drama television Query 1: drama single Query 2: drama film Query 3: drama arts members

QS1: Canon Products	
ISKR	Query 0:canonproducts:category:camera Query 1:canonproducts:category:camcorders Query 2:canonproducts:category:printer
PEBC	Query 0:canonproducts:category:camera Query 1:canonproducts:category:camcorders Query 2:canonproducts:category:printer
ICR	Query 0:canonproducts:category:camera Query 1:canonproducts:category:camcorders Query 2:canonproducts:category:printer
BQG	Query 0:canonproducts:category:camcorders Query 1:canonproducts:category:camera Query 2:canonproducts:category:printer
TFICF	Query 0:canonproducts:category:camera Query 1:canonproducts:category:camcorders Query 2:canonproducts:category:printer
Data Clouds	Query0: memory:category:flashmemory Query1: flashmemory:name:epsd/2gb Query2: flashmemory:name:sd
Google	Query0: “Olympus Products” Query1: “Canon Electronics” Query2: “Nikon Products”
F measure	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory Query 2:memory:category:ddr3

QS2: Networking	
ISKR	Query 0:networking products:category:routers Query 1:networking products:category:firewalls Query 2:switches:name:*switch* networking products:category:switches
PEBC	Query 0: networking products:category:routers Query 1: firewalls:name:*firewall * Query 2: networking products:category:switches
ICR	Query 1:networking products:category:switches Query 2:networking products:category:firewalls Query 3:networking products:category:routers
BQG	Query 0:networking products:category:firewalls Query 1:networking products:category:switches Query 2:networking products:category:routers
TFICF	Query 0: networking products:category:routers switches:leds:port Query 1: firewalls:name:*firewall* firewalls:throughput:mbps Query 2: switches:name:*hp * networking products:category:switches
F-measure	Query 0: networking products:category:routers Query 1: firewalls:name:*firewall * Query 2: networking switches:name:switch
Data Clouds	Query 0: firewalls:name:d-link Query 1: firewalls:vpn tunnels:8 Query 2: firewalls:name:dir-130
Google	q1: "Social Networking products" q2: "Computer Networking products" q3: "Networking products price

QS3: Routers	
ISKR	Query 0:routers:features:filtering routers:rj-5 ports:4 Query 1:routers:name:cisco * Query 2:routers:name:10/100 *
PEBC	Query 0:routers, routers:features:mac Query 1:routers, routers:name:cisco* Query 2:routers, routers:name:10/100 *
ICR	Query 1:routers, routers:name:10/100 * Query 2:routers, routers:name:band * routers:name:rangemax * Query 3:routers, routers:features:filtering
BQG	Query 0: routers, routers:device type:router Query 1: routers,routers:device type:vpn Query 2: routers,routers:device type:wireless
TFICF	Query 0: routers routers:name:rangemax dual* Query 1: routers routers:name:cisco* routers:name:integr* Query 2: routers routers:name:10/100 mbps*
F-measure	Query 0: routers:features:filtering routers:rj-5 ports:4 Query 1: routers routers:name:cisco * Query 2: routers routers:name:10/100 *
Data Clouds	Query0 : routers:name:lkr-604 Query1: routers:features:mac Query2: routers:name:broadband
Google	q1: "Networking, wireless, routers" q2: "Network, routers" q3: "Wood routers

QS4: TV	
ISKR	Query 0:tv:compatibility:720p Query 1:tv:resolution:1920 x1080
PEBC	Query 0:tv:compatibility:720p Query 1:tv:resolution:1920 x1080
ICR	Query 0:tv:compatibility:720p Query 1:tv:resolution:1920 x1080
BQG	Query 0: tv,tv:compatibility:720p Query 1: tv,tv:compatibility:1080p, tv:resolution:1920x1080 Query 2: tv,tv:compatibility:1080p,tv:outputs:optical
TFICF	Query 0: tv:compatibility:720p tv:resolution:1366x768 Query 1: tv:resolution:1920x1080
F measure	Query 0:tv:compatibility:720p Query 1:tv:resolution:1920 x1080
Data Clouds	Query0: tv:name:lcd* Query1: tv:name:26lg40* Query2:tv:outputs:audio
Google	q1: "TV, guide, products" q2: "TV, electronics" q3: "TV, hair products

QS5: TV Plasma	
ISKR	Query 0:tv:brand:panasonic Query 1:tv:brand:samsung tv:name:samsung *
PEBC	Query 0:tv:brand:panasonic Query 1:tv:brand:samsung
ICR	Query 1:tvplasma, tv:brand:panasonic Query 2:tvplasma, tv:brand:samsung tv:name:samsung*
BQG	Query 0: tv:resolution:1366x768 Query 1: tv:compatibility:1080p,tv:name:50ps60* Query 2: tvplasma,tv:compatibility:1080p,tv:name:plasma*
TFICF	Query 0: tvplasma tv:brand:panasonic tv:name:panasonic * Query 1: tvplasma tv:brand:samsung tv:name:samsung *
F-measure	Query 0:tv:brand:panasonic Query 1:tv:brand:samsung
Data Clouds	Query0: tv:condition:new Query1: tv:name:plasma Query2: tv:displaytype:plasma
Google	q1: "TV Plasma vs lcd" q2: "TV LCD" q3: "TV, bestbuy plasma

QS6: HP Products	
ISKR	Query 0:hpproducts:category:laptop Query 1:hpproducts:category:battery Query 2:hpproducts:category:printer
PEBC	Query 0:hpproducts:category:laptop Query 1:hpproducts:category:battery Query 2:hpproducts:category:printer
ICR	Query 1:hpproducts:category:laptop Query 2:hpproducts:category:battery Query 3:hpproducts:category:printer
BQG	Query 0:hpproducts:category:laptop Query 1:hpproducts:category:printer Query 2:hpproducts:category:battery
TFICF	Query 0: hpproducts:category:laptop laptop:platform:pc Query 1: hpproducts:category:battery battery:name:battery Query 2: hpproducts:category:printer printer:resolution:dpi
F measure	Query 0:hpproducts:category:laptop Query 1:hpproducts:category:battery Query 2:hpproducts:category:printer
Data Clouds	Query0: hpproducts:category:battery Query1: battery:name:hp Query2: battery:name:compaq
Google	q1: "HP Products Corporation" q2: "HP Printers" q3: "HP Laptops"

QS7: Memory	
ISKR	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory Query 2:memory:category:ddr3
PEBC	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory Query 2:memory:category:ddr3
ICR	Query 1:memory:category:harddrive Query 2:memory:category:flashmemory Query 3:memory:category:ddr3
BQG	Query 0:memory:category:flashmemory Query 1:memory:category:harddrive Query 2:memory:category:ddr3
TFICF	Query 0: memory memory:category:harddrive harddrive:dimensions:x Query 1: memory memory:category:flashmemory flashmemory:name:card Query 2: memory memory:category:ddr3 ddr3:memory category:desktop
F measure	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory Query 2:memory:category:ddr3
Data Clouds	Query0: memory:category:flashmemory Query1: flashmemory:name:epsd/2gb Query2: flashmemory:name:sd
Google	q1: "Human memory" q2: "Computer memory" q3: "Memory game

QS8: Memory 8GB	
ISKR	Query 0:memory:category:harddrive Query 1:memory:category:ddr3 Query 2:memory:category:flashmemory
PEBC	Query 0:memory:category:harddrive Query 1:memory:category:ddr3 Query 2:memory:category:flashmemory
ICR	Query 1:memory:category:harddrive Query 2:memory:category:ddr3 Query 3:memory:category:flashmemory
BQG	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory Query 2:memory:category:ddr3
TFICF	Query 0: Memory:category:harddrive harddrive:capacity:8gb Query 1: Memory:category:ddr3 ddr3:memorysize:8gb Query 2: Memory:category:flashmemory flashmemory:memorysize:8gb
F measure	Query 0:memory:category:harddrive Query 1:memory:category:ddr3 Query 2:memory:category:flashmemory
Data Clouds	Query0: memory:category:ddr3 Query1: ddr3:memorysize:8gb Query 2:memory:category:flashmemory
Google	q1: "Memory cards 8gb" q2: "Laptop memory, 8GB" q3: "Flash memory

QS9: Memory Internal	
ISKR	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory
PEBC	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory
ICR	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory
BQG	Query 0: memory:category:harddrive Query 1: memory:category:flashmemory, flashmemory:name:atech* Query 2: memory:category:flashmemory, flashmemory:name:internal*
TFICF	Query 0: harddrive:drivetype:internal memory:category:harddrive Query 1: memory:category:flashmemory flashmemory:name:internal
F measure	Query 0:memory:category:harddrive Query 1:memory:category:flashmemory
Data Clouds	Query0: flashmemory:name:xm-5u Query1: flashmemory:name:pro-gear
Google	q1: "dell internal memory" q2: "d internal dell

QS10: Printer	
ISKR	Query 0:printer:resolution:1200 dpi Query 1:printer:resolution:600 dpi
PEBC	Query 0:printer, printer:resolution:1200 dpi Query 1:printer, printer:resolution:600 dpi
ICR	Query 1:printer, printer:resolution:600 dpi Query 2:printer, printer:resolution:1200 dpi
BQG	Query 0: printer:printmethod:inkjet Query 1: printer:printmethod:laser, hpproducts:category:printer Query 2: printer:printmethod:laser, printer:name:q7816a
TFICF	Query 0: printer printer:resolution:1200 dpi printer:resolution:optimized Query 1: printer printer:paperinput:150 hpproducts:category:printer
F measure	Query 0:printer:resolution:1200 dpi Query 1:printer:resolution:600 dpi
Data Clouds	Query0: printer:resolution:(color) Query1: printer:networkready:no
Google	q1: "Canon, Printer" q2: "HP, Printer