Exploiting Underrepresented Query Aspects for Automatic Query Expansion

Daniel Crabtree
daniel@danielcrabtree.com

Peter Andreae
pondy@mcs.vuw.ac.nz

Xiaoying Gao
xgao@mcs.vuw.ac.nz

School of Mathematics, Statistics and Computer Science
Victoria University of Wellington
New Zealand

ABSTRACT

Users attempt to express their search goals through web search queries. When a search goal has multiple components or aspects, documents that represent all the aspects are likely to be more relevant than those that only represent some aspects. Current web search engines often produce result sets whose top ranking documents represent only a subset of the query aspects. By expanding the query using the right keywords, the search engine can find documents that represent more query aspects and performance improves. This paper describes AbraQ, an approach for automatically finding the right keywords to expand the query. AbraQ identifies the aspects in the query, identifies which aspects are underrepresented in the result set of the original query, and finally, for any particularly underrepresented aspect, identifies keywords that would enhance that aspect's representation and automatically expands the query using the best one. The paper presents experiments that show AbraQ significantly increases the precision of hard queries, whereas traditional automatic query expansion techniques have not improved precision. AbraQ also compared favourably against a range of interactive query expansion techniques that require user involvement including clustering, web-log analysis, relevance feedback, and pseudo relevance feedback.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*query formulation*; I.2.7 [**Artificial Intelligence**]: Natural Language Processing—*text analysis*

General Terms

Algorithms, Experimentation, Performance

Keywords

query expansion, web search, global document analysis, aspect coverage

KDD'07, August 12–15, 2007, San Jose, California, USA.

1. INTRODUCTION

Users approach web search engines to find web pages that satisfy their search goals. They specify a query to capture the intent of their search goal and in response, the search engine returns a result set containing the best matching pages from its corpus. If the query does not express their search goal well enough, the result set will contain few useful pages, and the users have to refine their query to capture the search goal more effectively. How the query should be refined varies, depending on the reason why the query did not express the search goal.

Query refinement is a time consuming process for users. Users must look through the retrieved pages to identify the irrelevant pages and then they must work out how to modify their query to filter out the irrelevant results. Fortunately, there are many opportunities for the search system to help the user refine their query. Automatic query expansion is an approach for improving query performance with no user involvement by automatically modifying the user's query. Automatic query expansion works by addressing failures of the search mechanism. One such failure is that search engines only find pages that include the query words. Many relevant pages may use similar words, but not exactly those used in the query and so those relevant pages will not be in the result set. Automatic query expansion can help identify the other relevant pages by identifying additional terms that occur on relevant pages. These additional terms can be identified by examining the pages that have been retrieved by the search engine.

There are many methods of automatic query expansion based on pseudo, blind, or ad-hoc relevance feedback. The methods work by assuming some initial portion of the result set is relevant and then proceeding with standard relevance feedback methods [21]. The methods vary in several ways: the retrieval model used, how terms are weighted, and how to select terms. There are three common retrieval models: the Boolean model, the vector space model, and the probabilistic model. The Boolean methods are relatively simple in their use of weighting, effectively they are limited to adding additional query terms [11] and modifying the Boolean connectives (AND and OR) between query terms [14]. The most common term weighting method for the vector space model is Rocchio's method [20, 3]. Robertson and Sparck Jones introduced four term weighting schemes for probabilistic models [19, 25]. Another weighting method for the probabilistic model is the Bayesian approach [25].

Most research on query expansion has focused on opti-

mizing the incorporation of additional terms through the retrieval model and term weighting methods. However, the quality of the additional terms is also very important, yet there is little research on term selection, as evidenced by the limited techniques for Boolean models. Terms are typically selected using local document analysis from the top N ranked documents in the initial result set and the terms are typically ranked using document frequency (df), term frequency (tf), or term frequency inverse document frequency (tfidf) [24]. Local document analysis has the problem that performance only improves when the top ranking documents are relevant. Terms can also be selected by global document analysis by looking for words that frequently co-occur with query terms in the whole corpus [26] or using thesauri like Word-Net [12]. A problem with using global document analysis or thesauri based approaches is that query drift (changing the meaning of the query) increases, hurting precision. This paper investigates improving the quality of term selection, rather than on improving the retrieval model and term weighting methods, and therefore the relatively simple Boolean model without term weighting is used.

Search goals are frequently a composition of several aspects. For example, a search goal of finding possible holidays would have one aspect, but searching for travel agents in Los Angeles who deal with cruises involves three different aspects. To express their search goal as a query, users typically list one or more words for each aspect from the search goal. The search goal aspects that are explicitly represented in the query in this way are the query aspects.

A significant problem with web search that requires users to refine their queries is the aspect coverage problem [4]: the documents do not cover all the different query aspects. In general, the more aspects a query has, the fewer documents there are in the result set that cover all the aspects. Current search systems frequently return biased result sets with documents that focus on one aspect of the search goal and let other aspects take a subsidiary role or focus on an irrelevant aspect [2]. To deal with this problem, query refinement methods could choose refinements that increase aspect coverage of queries. A benefit of this approach is that the refinements are generated independently of whether the original query retrieves relevant documents.

While existing automatic query expansion methods successfully increase recall for easy queries with high precision, they provide little benefit for hard queries with low precision. This paper investigates and develops an automatic query expansion method to improve the precision of hard queries by finding terms that address query aspects that are underrepresented in the result set.

After outlining the background to automatic query refinement, the paper discusses the problems posed by multi-aspect queries for current web search engines. Section 4 introduces a new automatic query expansion method, AbraQ, which identifies query aspects, identifies underrepresented aspects that are missing from the result set, and then chooses a suitable refinement term to add to the query. In section 5, AbraQ is evaluated and the results show that AbraQ significantly improves the precision of hard queries on which the underlying search system has poor performance. The evaluation also shows that AbraQ not only outperforms other automatic query expansion methods, but performs comparably with the optimal performance of many good interactive query expansion methods that depend on feedback from the user, including clustering, web-log analysis, relevance feedback, and pseudo relevance feedback.

2. BACKGROUND

Besides automatic query expansion, an alternative approach for improving query performance is interactive query expansion, which involves user input, either in the form of relevance judgments about documents as in relevance feedback or through selection of refinement terms suggested by the system [21]. Relevance feedback methods are identical to those used for automatic query expansion, except they are based on the relevant documents specified by the user, instead of the top ranked documents. Two popular techniques for generating refinements are web page clustering [7] and query log analysis [9]. While the focus of this paper is on automatic query expansion, it is useful to consider interactive query expansion methods too as they use similar techniques and they offer a nice platform for performance comparison as in some sense they make optimal refinements.

Automatic query expansion systems have typically tried to enhance recall and not precision [24]. On easy queries where the initially retrieved documents are already quite relevant, recall is improved. On hard queries where the initially retrieved documents are irrelevant, the modifications do not help. Queries are typically extended with 20 or more terms [21], which often helps recall, but hurts precision due to query drift. With web search, where the corpus is much larger, the problem is to find some relevant documents; upon finding the right keywords, there are usually more than enough relevant documents. Precision is therefore more important than recall. This paper considers methods that make smaller query modifications, suitable for precision enhancement and web search.

Both local (using only documents in the result set) and global document analysis (using thesauri or properties of the entire corpus) provide sources for refinement terms [26]. The association hypothesis states that "if an index term is good at discriminating relevant from irrelevant documents, then any closely associated index term is also likely to be good at this." [23]. Early attempts to apply the association hypothesis using co-occurrence information to query refinement had poor results [21]. But recently, due to much larger corpora, co-occurrence information has become more reliable as shown by the Google similarity distance [5]. This kind of global document analysis has been applied to web page clustering and was shown to significantly improve performance over algorithms using only local document analysis [7]. AbraQ also uses global document analysis in the form of search result counts to help identify new refinement terms, but uses the counts in several new ways.

One part of AbraQ involves analyzing the query to identify aspects. Previously, researchers have identified common phrases in queries and used these to improve performance. Structured queries are Boolean queries containing AND, OR, NOT and PROXIMITY operators like phrases. Structured queries can improve retrieval performance [8], but users rarely use structured queries and tend to prefer unstructured and natural language queries [13]. Two approaches to query refinement attempt to identify phrases for structured queries from natural language queries: the syntactic tagging approach which uses part of speech tagging and the dictionary approach [8]. The problem is that many web search queries are not natural language, but just

an unstructured sequence of words. While dictionary approaches work with unstructured queries, they are limited in their scope and usually only work with domain specific corpora.

Other researchers have also tried to identify aspects and have termed the process query splitting. Query splitting is the process of splitting the query into its individual aspects. A trivial approach is to split the query into single words [27], although this does not sufficiently capture the aspects as many words change substantially from their individual meaning once placed into a sequence. A more sophisticated approach uses clustering to identify multi-word aspects [28]. We are proposing a new approach that uses global document analysis to identify query aspects.

One relatively new research area is estimating query difficulty. One suggested benefit of estimating query difficulty is that it could improve automatic query expansion by identifying easy queries and applying automatic query expansion to only those easy queries [27]. However, easy queries are less likely to need refinement. In this paper, we show how to improve automatic query expansion for hard queries by identifying underrepresented query aspects and present a successful method of distinguishing between easy and hard queries.

Previous research has found that aspects affect query expansion. Terms that reflect multiple aspects of the original query are often better [26, 6]. Researchers have also found analyzing aspects to be useful in other areas. Another approach that uses aspects is the stepping stones and pathways method [17], but it is solving a different problem — where there is no single document that contains all the desired content and the desired content can be found from a series of documents spread through a sequence of connected but separate searches. We propose a new automatic query expansion method that identifies the aspects of an unstructured query, identifies which aspects are underrepresented in the result set, and finds a suitable refinement.

3. MULTI-ASPECT QUERIES

Constructing an effective query for a search goal involves finding a conjunction of terms that co-occur in documents that satisfy the search goal. The terms must be both distinctive and discriminating: they must occur in many of the desired documents, while not appearing together in undesired documents.

Frequently, many documents contain all the query terms, but just focus on a subset of the query aspects. For example, the two aspect query "black bear attacks" may produce a result set consisting of documents that discuss black bears in detail, but only mention in passing that they sometimes attack. Although this result set contains all desired terms, it only focuses on the "black bear" aspect, and has missed the "attacks" aspect.

There are different approaches to refining queries with underrepesented aspects. Consider the search goal of finding documents that discuss the effectiveness of increased airport security. A typical starting query could be "airport security". However, the result set might focus on the security aspect and contain documents for security company websites and financial news articles about those companies. One refinement approach is to add additional descriptive words. These could be for search goal aspects that are not yet in the query or for existing query aspects. A search goal aspect could be introduced into the query by adding the word "effectiveness" to the query, but that may not help with the initial problem; in fact, it may even narrow the results further in the wrong direction, towards the company news articles discussing the effectiveness of some new procedure. An alternative would be to add an additional descriptive word such as "airfield" for the underrepresented airport aspect. While that would probably ensure the airport aspect was well represented, it would also probably shift the other query aspects out of focus, and it may also cause query drift, or eliminate many relevant documents.

Web search systems work by finding documents that contain terms whether or not the terms are descriptive of the document contents. When expanding a query for refinement purposes, users often choose additional descriptive terms because it is natural for them to think descriptively about search goals, but such terms are often ineffective because the terms that describe the content are not necessarily present in the documents and may not be very discriminating. On the other hand, search engines easily cope with indirectly related terms that co-occur frequently with other descriptive terms, these terms are often effective because the terms are distinctive and discriminating. Our idea is to refine queries with underrepresented aspects using indirect refinements. Indirect refinements use co-occurring words that are not necessarily descriptive words. For example, in the previous airport security example, adding the word "baggage" or "terrorist" may bring the airport aspect into the documents without causing query drift or eliminating many relevant documents.

While it is incredibly hard for users to identify indirect refinement terms, AbraQ[1] can provide these kinds of refinement by leveraging global document analysis techniques. The problem are how to identify the query aspects automatically, how to identify which query aspects are underpresented in the result set automatically, and then how to pick the best indirect refinement term for addressing the underrepresented aspect.

4. ALGORITHM

AbraQ is an automatic query expansion method that uses local and global document analysis. The algorithm has three steps: identify query aspects, identify underrepresented aspects, and identify the refinement. This section is broken into two main parts: 4.1 describes, at a higher level of abstraction, the key ideas and principles behind AbraQ; 4.2 describes our current implementation of these ideas and principles and this is the implementation of AbraQ that is used for evaluation.

4.1 Ideas and Method

4.1.1 Identify Query Aspects

The order of words in unstructured queries carries important semantic information: users typically group words related to a single aspect into phrases. For example, users may search for "Microsoft office 2007 reviews" or "reviews Microsoft office 2007", but they probably would not search for "Microsoft reviews 2007 office" or "2007 office reviews Microsoft". Finding sub-sequences of the query words that

[1]AbraQ identified both "terrorist" and "baggage" as high ranking refinement terms for the query "airport security"

commonly occur together as a phrase will help identify query aspects.

Global document analysis provides two factors that identify aspects: existence and support. To be an aspect, the sub-sequence of words must *exist* — occur frequently enough relative to the frequency of the set of words, and have *support* — occur frequently enough relative to the frequency of all other permutations of the same words.

Any subsequence of the query could represent an aspect: bi-grams, tri-grams, and so on, up to n-grams, where n is the length of the query. We can reduce the number of subsequences that have to be considered by constructing subsequences greedily from left to right and making the assumption that if an i-gram subsequence is not an aspect, then it cannot be extended to become an $(i+1)$-gram aspect. This reduces the number of aspect checks for an n word query from $O(n^2)$ to $O(n)$.

4.1.2 Identify Underrepresented Aspects

Different aspects invoke different vocabulary. For example, "Microsoft office 2007" may invoke terms like "windows", "spreadsheet", and "word processor", while "review" may invoke terms like "compare", "performance", and "evaluate". Documents that reflect an aspect are more likely to use at least some of the vocabulary related to that aspect. An aspect is probably underrepresented in a result set if the documents do not sufficiently represent the vocabulary of the aspect.

The vocabulary of each aspect can be constructed by analysing the frequencies of terms in documents returned from all the sub-queries consisting of a subset of the aspects of the full query. Sub-queries containing fewer aspects are more likely to return documents containing the vocabulary of the aspects. Any sub-query containing an aspect can contribute to that aspects vocabulary, but those with fewer aspects should be weighted more heavily.

If the documents from the original query do not contain enough of the vocabulary model of an aspect, then that aspect is potentially underrepresented. For example, in the example above, the words "compare" and "performance", which are associated with "review" may not be present in many documents from the result set, indicating that the review aspect may be underrepresented. The aspect score is a measure of the relative degree to which an aspect is represented or underrepresented, and can be computed from the term frequencies in the documents from the original query and the vocabulary models.

4.1.3 Identify Refinement

If some aspects are underrepresented in the original result set, then a good refinement is a query that produces a result set with no underrepresented aspects. Therefore, terms that are strongly related to underrepresented aspects are good candidates for expanding the original query, as they are more likely to lead to good refinements. Additionally, since all query aspects should be represented in the result set, terms that are related to multiple aspects are better than terms that relate to only one aspect.

The quality of the refinement can be measured using the aspect scores of the refined query.

4.2 Prototype Implementation Details

The following section details the current implementation of AbraQ. We have made no attempt to tune parameters or optimise the algorithms.

4.2.1 Identify Query Aspects

To identify aspects of the query, the current implementation of AbraQ does a greedy search for subsequences of the query that have a score greater than a predefined threshold. AbraQ defines the score of a subsequence as the product of the *Existence* and *Support* factors for the subsequence. *Existence* and *Support* are defined in terms of $D(s)$, the number of documents that contain each of the words in a sequence s, and $DP(s)$, the number of documents that contain s as a phrase.

$$Existence(s) = \frac{DP(s)}{D(s)}$$

$$Support(s) = \frac{DP(s)}{\sum_{s' \in \mathrm{Perm}(s) \setminus \{s\}} DP(s')}$$

where $\mathrm{Perm}(s)\setminus\{s\}$ is the set of all permutations of the sequence s other than s itself.

AbraQ currently considers a subsequence of the query to be an aspect if $Existence(s) \cdot Support(s) \geq 1.0$. Alternatively, instead of producing a trade-off between the factors, an implementation could consider each factor separately. Future work could investigate the difference between these approaches.

If two parts of the query both describe the same aspect, this implementation identifies multiple synonymous aspects. This poses no problem for this application, but it may be possible to improve performance or efficiency by identifying this relationship between aspects. Future work could explore using a thesaurus like WordNet or other kinds of global document analysis to identify synonymous aspects.

If there is only one query aspect in the query then the result set of the original query is presented to the user and AbraQ makes no refinement.

4.2.2 Identify Underrepresented Aspects

To construct vocabulary models of each aspect, AbraQ runs sub-queries for all aspects and all pairs of aspects identified in the first step, and finds all the terms that occur in the documents returned. (The current implementation restricts the result set to just the first 10 documents, but retrieving more documents would improve the statistics.) For each aspect, it ranks the terms according to document frequency (the number of retrieved documents containing the word from the sub-queries that contained the aspect), retaining the top 200 terms only. Then, for each aspect, it adds to the vocabulary model of the aspect, the 50 best terms from the top 200, ranked according to the strength of co-occurrence with the aspect.

The sub-queries currently use the set of words associated with each aspect rather than the phrase of the sequence of words; future work could explore the restriction of the queries to the phrases.

The co-occurrence strength ($CS(t, a)$) is the ratio between the actual and the expected co-occurrence frequency of a term t and an aspect a. The actual co-occurrence frequency is the fraction of documents that contain both the aspect and the term. The expected co-occurrence frequency is calculated by assuming that the aspect and the term are independent and computing the product of the fraction of docu-

ments that contain the aspect and the fraction of documents that contain the term.

$$CS(t,a) = \frac{D(t \wedge a) \cdot D}{D(a) \cdot D(t)}$$

The vocabulary model ($Vocab(a)$) of an aspect is a normalized weighted vector of terms that had a high co-occurrence with the aspect. The term weights are the sum of the contributions to each term from the sub-queries. The contribution is the co-occurence strength divided by the number of aspects in the sub-query. AbraQ normalizes each model's term weights to one using an independence assumption similar to Naive Bayes, treating the weightings from multi-aspect queries as independent of the weightings from single-aspect queries. Future work could consider a richer Bayesian model.

$$Vocab(a) = \{\{t_1, w_1\}\{t_2, w_2\} \cdots \{t_n, w_n\}\}$$

$$w_i = \frac{1}{N} \sum_{q' \in sub\text{-}queries} \frac{CS(t_i, a)}{|q'|}$$

where N is the normalizing factor to ensure $\sum_i (w_i) = 1$ and $|q'|$ is the number of aspects in the sub-query q'.

AbraQ identifies underrepresented aspects by scoring the vocabulary model of each aspect against the documents from the original query. The raw aspect score ($RAW(a,q)$) is the dot product of the weights in the aspect's vocabulary model and the term frequencies in the documents returned by the original query q. The relative aspect scores ($RAS(a,q)$) are the raw aspect scores normalized so they sum to one, and indicate the relative probabilities that aspects are underrepresented.

An aspect is considered underrepresented if its relative aspect score is below the representation level threshold (RLT), which depends on the number of aspects ($|q|$). For queries with two aspects, the representation level threshold is 33%; for queries with three aspects, it is 25%.

$$RLT(q) = \frac{1}{|q| + 1}$$

$$RAS(a,q) < RLT(q) \longrightarrow a \text{ is underrepresented}$$

If an aspect is very underrepresented, then this constitutes evidence that it may not be a valid aspect at all. If the relative aspect score $RAS(a,q)$ of an aspect is below a minimum threshold of 20% of the $RLT(q)$, then AbraQ backs off its assumption that the terms form an aspect and (if possible) tries splitting the aspect into two sub-aspects. AbraQ splits the aspect by removing the last word from it and forming a new aspect using this word. AbraQ then recalculates the aspect scores with the new set of aspects. It continues splitting very underrepresented aspects until either they are single words or the aspect score is greater than the minimum threshold. This back-off procedure ensures that misidentified aspects from the first stage of the algorithm do not cause the system to misidentify an underrepresented aspect.

4.2.3 Identify Refinement

If there are no underrepresented aspects then the result set does not suffer from the aspect coverage problem, and AbraQ does not attempt to refine the query. If there are underrepresented aspects, then AbraQ picks the aspect with the worst relative aspect score and tries to identify a refinement to improve the representation of that aspect.

The vocabulary models in the previous stage used term weightings based on the co-occurrence strength of the term with the aspect. AbraQ identifies candidate terms by selecting the higher weighted terms from the vocabulary model of the least represented aspect. For each possible term, AbraQ constructs a new query consisting of the old query plus the refinement term, runs the new query, then re-computes all the aspect scores (as in the previous stage) but using the result set of the refined query instead of the original query. To score a refinement q', it sums the new aspect scores, weighting the previously underrepresented aspects more heavily.

$$RS(q') = \sum_a \frac{RAW(a,q')}{RAS(a,q)}$$

where the sum is taken over the aspects a in the original query.

AbraQ then presents the result set of the highest scoring query refinement to the user.

5. EVALUATION

We evaluated AbraQ by comparing its performance to a set of automatic query expansion methods and a set of interactive query expansion methods. We applied all the algorithms to a set of hard queries and measured their effectiveness at improving the precision of the queries on Google.

5.1 Testing Set

It is most important to evaluate refinement performance on hard queries, as easy queries already have good performance and do not require refinement. The testing queries are the topic titles of ten queries from the TREC 2005 hard track (topic numbers: 303, 307, 310, 314, 322, 325, 336, 341, 363, and 416). We selected these test queries by taking the first one or two topics (by topic number) from each of the seven types of queries that were identified by [30] in the TREC 2005 hard track. The queries had between one and three aspects and varied in length between two and five words. The mean query length is 2.9, which is similar to the typical user query of length 2.92 [13]. Some of the queries are hard: three had no relevant results and three others had few relevant results in the initial Google search. Some result sets have underrepresented aspects and some focus on irrelevant aspects.

5.2 Measurements

The results were evaluated using two measurements: P@5 and P@10, which are the precision of the first 5 and first 10 documents in the search results respectively, where precision [10] is the number of relevant documents retrieved divided by the number of documents retrieved (5 or 10 in this case). P@n measures do have problems when there are not enough relevant documents in the collection [22], but this is not a problem in our case as there are more than enough relevant documents for the search tasks used in our experiments. The reason for evaluating precision using just the first 5 and 10 documents is because most of the time (over 70%) web search users only look at the first page of results [13]. The relevancy judgements were based on the description and narrative from the TREC 2005 hard track. The description specifies the user's search goal and the narrative gives specific details on what would be relevant and irrelevant documents.

Table 1: Automatic query expansion methods compared on 10 search tasks from TREC 2005 hard track against Google baseline

Method	Precision (Std Dev)	Queries Improved	Queries Worsened
First five results			
GOOGLE	40% (35%)	-	-
AbraQ	**62% (20%)**	**50%**	**0%**
df1	36% (39%)	20%	30%
df5	40% (33%)	20%	20%
tf1	32% (30%)	10%	40%
tf5	38% (29%)	20%	20%
tfidf1	32% (32%)	10%	30%
tfidf5	42% (38%)	20%	30%
First ten results			
GOOGLE	38% (30%)	-	-
AbraQ	**52% (22%)**	**60%**	**0%**
df1	32% (27%)	10%	30%
df5	35% (35%)	20%	20%
tf1	29% (28%)	0%	30%
tf5	40% (35%)	30%	20%
tfidf1	34% (32%)	20%	30%
tfidf5	37% (36%)	20%	20%

5.3 Automatic Query Expansion

AbraQ provides automatic query expansion, which means that it directly presents the user with a result set and takes no input from the user beyond their original query. We compared AbraQ against six different automatic refinement methods. All methods could benefit equally from more sophisticated document models and term weighting; however the focus is on the term selection component, so the simple Boolean model is used for evaluation, as this excludes the effect of term weighting. As in [24], the three term ranking methods used were document frequency (df), term frequency (tf), and term frequency inverse document frequency (tfidf). The two selection methods used to select the terms for query expansion were the top term (1), and the disjunction of the top five terms (5). This gave six methods (df1, df5, tf1, tf5, tfidf1, and tfidf5) for automatic query expansion and all used terms from the top five documents.

Our results shown in table 1 show that traditional automatic query expansion methods have a negative impact on precision and this agrees with past findings that these methods focus on recall enhancement and not precision [24]. Other researchers have found that around 25% of queries suffer from query drift [21] leading to lower precision. Our findings corroborate this with just over 26% of queries having worse precision after expansion using one of the six traditional methods. The results also seem to indicate that using fewer terms for query expansion is more likely to cause query drift in Boolean queries. On closer examination of the results, queries where the result set initially had no relevant documents were mostly unaffected and those that had relevant documents initially were affected slightly positively or negatively about an equal number of times, but sometimes a significant drop in precision occurred due to query drift. At best, the traditional automatic query expansion methods have little effect on precision and may reduce it. In contrast, AbraQ provided substantial improvement to many queries without decreasing performance of any others.

The improvement made by AbraQ was significant. Significance was tested by comparing each pair of methods using the Wilcoxon signed-rank test. The results showed that there was no significant difference between Google and any of the six automatic query expansion methods, but AbraQ was significantly better at a 95% level of confidence than Google and each of the other six methods.

5.4 Detailed Analysis

A more detailed analysis of the ten queries is shown in table 2. AbraQ modified six of the ten queries and within the first five results, AbraQ improved five of the six queries it modified, and improved all six queries within the first ten results. The four queries that were not modified by AbraQ, were the queries for which Google provided good results. Google's precision for the first five documents on those four queries was 70%, compared to 20% for the other six queries that AbraQ modified. So AbraQ significantly improves the performance of queries that are hard for the underlying search system and does not affect queries that are easy for the underlying search system. The benefit of improving hard query performance is even greater than is initially apparent: improving low precision queries can mean the difference between satisfying the search goal or not, whereas high precision queries already provide sufficient answers and so the benefit of improvement is less for high precision queries.

Table 2 shows that AbraQ only affected queries with poor initial performance on Google. This shows that identifying the underrepresented aspects using AbraQ provides an accurate method of identifying these poor performing queries. It also suggests AbraQ could provide a method for estimating query difficulty which has the benefit of making this technique applicable to other applications [27].

The results show that addressing poor aspect coverage can significantly improve many hard queries, lending credence to the suggestion by other researchers [4] that the primary cause of poor search results in multi-aspect queries is poor aspect coverage.

5.4.1 Aspect Identification

There is no rule about what constitutes an aspect: for one search, a sequence of words may be an aspect, but for a different search, the same word sequence may form two or more aspects. The backing off in the second stage extends the search of the query aspect space from what was done greedily with the initial construction. The additional searching takes place using guidance from the result set to constrain the search to the right part of the search space.

As expected, AbraQ is overly zealous at joining words together into aspects in the first step of the algorithm. After the first step, in eight of the ten queries, the query aspects identified matched the actual aspects of the search goal. In both cases where the aspects did not match, the aspects identified had joined too many words together. Fortunately, this was expected and it did not affect performance as the back-off procedure in the second step correctly reduced the aspects in these two cases to match the correct aspects.

5.4.2 Identify Underrepresented Aspects

The method correctly identifies the underrepresented as-

Table 2: Segmented results comparing Google and AbraQ 10 search tasks from TREC 2005 hard track

Method	Precision (Std Dev)	Queries Improved	Queries Worsened
First five results			
All Queries			
GOOGLE	40% (35%)	-	-
AbraQ	**62% (20%)**	**50%**	**0%**
4 unmodified queries: 303, 307, 310, 416			
GOOGLE	70% (12%)	-	-
AbraQ	**70% (12%)**	**0%**	**0%**
6 modified queries: 314, 322, 325, 336, 341, 363			
GOOGLE	20% (31%)	-	-
AbraQ	**57% (23%)**	**83%**	**0%**
First ten results			
All Queries			
GOOGLE	38% (30%)	-	-
AbraQ	**52% (22%)**	**60%**	**0%**
4 unmodified queries: 303, 307, 310, 416			
GOOGLE	68% (13%)	-	-
AbraQ	**68% (13%)**	**0%**	**0%**
6 modified queries: 314, 322, 325, 336, 341, 363			
GOOGLE	18% (18%)	-	-
AbraQ	**42% (21%)**	**100%**	**0%**

pects and is able to identify the cases where all aspects are sufficiently represented.

In five of the six cases where AbraQ performed query modification, the aspects that appeared on inspection to be underrepresented in the results agreed with those identified by AbraQ. So in most cases, AbraQ identifies the underrepresented aspects correctly. The five cases were the same ones that AbraQ improved performance in the first five results.

In the remaining case where AbraQ performed query modification, an aspect that appeared to be represented was identified as underrepresented. However, in the first five results performance was not affected and in the first ten results performance improved. This was expected, as in these cases, AbraQ selects similar terms to those selected by traditional query expansion methods, and those methods have negligible effect on performance. Therefore, when the second step of AbraQ fails, performance should be no worse than the underlying search system. Yet, AbraQ may still improve the search as in selecting the best refinement term, it looks at all aspects, not just the one it believes is most underrepresented. Therefore, even if it is wrong about an aspect being underrepresented, the refinement terms will probably improve the representation of other aspects. More experiments that dealt with failure cases would be required to validate this hypothesis.

In the four cases where AbraQ did not perform query modification, there was either just a single aspect (one out of the four cases) and therefore trivially well represented in the result set or all aspects were well represented in the result set (three out of the four cases). Note that AbraQ trivially ignores single aspect cases with a single word, and that is why there was no need to test any single word queries.

Table 3: AbraQ and interactive query refinement methods compared on 10 search tasks from TREC 2005 hard track against Google baseline

Method	5 results Precision (Std Dev)	5 results Queries Improved	10 results Precision (Std Dev)	10 results Queries Improved
All Queries				
GOOGLE	40% (35%)	-	38% (30%)	-
AbraQ	**62% (20%)**	**50%**	**52% (22%)**	**60%**
Optimal	**82% (18%)**	**90%**	**71% (21%)**	**100%**
STC	48% (34%)	40%	44% (34%)	50%
LINGO	60% (27%)	50%	54% (25%)	50%
QDC	64% (23%)	60%	56% (24%)	60%
MAMMA	42% (38%)	10%	43% (37%)	20%
RAQE	48% (39%)	20%	45% (35%)	30%
PRIQE	54% (40%)	40%	50% (32%)	50%
4 unmodified queries: 303, 307, 310, 416				
GOOGLE	70% (12%)	-	68% (13%)	-
AbraQ	**70% (12%)**	**0%**	**68% (13%)**	**0%**
Optimal	**90% (12%)**	**100%**	**85% (13%)**	**100%**
STC	75% (19%)	25%	80% (8%)	50%
LINGO	75% (19%)	25%	75% (13%)	25%
QDC	80% (16%)	50%	80% (12%)	50%
MAMMA	75% (19%)	25%	80% (12%)	50%
RAQE	75% (19%)	25%	80% (14%)	50%
PRIQE	80% (23%)	50%	80% (8%)	50%
6 modified queries: 314, 322, 325, 336, 341, 363				
GOOGLE	20% (31%)	-	18% (18%)	-
AbraQ	**57% (23%)**	**83%**	**42% (21%)**	**100%**
Optimal	**77% (20%)**	**83%**	**62% (21%)**	**100%**
STC	30% (30%)	50%	20% (17%)	50%
LINGO	50% (28%)	67%	40% (21%)	67%
QDC	53% (21%)	67%	40% (13%)	67%
MAMMA	20% (31%)	0%	18% (24%)	0%
RAQE	30% (40%)	17%	22% (20%)	17%
PRIQE	37% (41%)	33%	30% (24%)	50%

5.5 Interactive Query Refinement

In addition to the automatic query modification methods, there are many methods of interactive query modification. These methods are quite different from AbraQ, in that they depend on additional information from the user. While these methods do not compete with AbraQ and could be used in combination with AbraQ, we include them here to test the upper limits of AbraQ.

The one-step refinement performance of a representative range of interactive query refinement methods were compared with AbraQ. For each interactive refinement method, the results shown in table 3 are the optimal performance, where a perfect user selects optimally from the fifteen suggestions made by these methods or chooses to make no refinement. Table 3 also shows the performance of AbraQ and the performance of Optimal AbraQ (Optimal), which corresponds to the optimal expansion selection from the top fifteen ranked AbraQ expansions.

5.5.1 Summary of Approaches

The interactive query refinement methods include three clustering algorithms (Suffix Tree Clustering (STC) [29],

Lingo [18], and Query Directed Clustering (QDC) [7]), a query log analysis method (Mamma search engine [16]), and two query expansion methods [21], one based on document relevance feedback (RAQE), and one that lets users select from the top ranking tfidf terms from the initial result set (PRIQE).

The clustering approaches find groups of similar pages in the result set and present these as refinements. All the standard data clustering methods [1] have been applied to web page clustering: hierarchical (agglomerative and divisive), partitioning (probabilistic, k-medoids, k-means), Bayesian, and many more. For comparison, we have used the most successful clustering approaches which use web or document specific characteristics to assist clustering: Suffix Tree Clustering (STC) [29] and Lingo [18] use phrases and Query Directed Clustering (QDC) [7] uses the relationship between terms and the query.

Query log analysis [9] finds similar queries through either textual similarity or from their relative position in a sequence of queries during the search sessions of different users. The most similar queries are presented as refinements. Effective query log analysis requires having sufficiently large query logs and so we chose to compare against the Mamma search engine [16].

Relevance feedback methods are similar to the methods of automatic query expansion discussed earlier. During document relevance feedback as in RAQE, the user specifies which documents in the initial result set are relevant, as opposed to the top N documents being assumed relevant, as in automatic query expansion. Then the terms from the relevant documents that do not appear in the irrelevant documents are ranked using tfidf and the best term is selected for refinement.

Pseudo, blind, or ad-hoc relevance feedback methods are identical to automatic query expansion. However, instead of simply picking the best term for refinement, the top ranked terms are presented to the user as in PRIQE.

5.5.2 Results Discussion

Table 3 shows that AbraQ performs well, even against methods that require further user input and receive the best possible input. This user input provides additional insight into the search goal and therefore can potentially improve the four unmodified queries. Over all ten queries, AbraQ outperforms all other methods except one in the first five results and two in the first ten results. On the six modified queries, AbraQ outperformed all other methods within both the first five results and the first ten results. In all cases, when AbraQ is on level footing with the other algorithms, and the optimal refinement selection is made from fifteen possible refinements then AbraQ labeled Optimal clearly outperforms the other interactive methods of query refinement.

Optimal AbraQ is able to improve the performance of the four queries that AbraQ did not try to modify and Optimal AbraQ improves upon all but one of AbraQ's six modifications. This was expected, as different refinement terms hint at subtly different relationships between the aspects and since the system is not privy to the information encoded in the user search goal, it is unable to achieve performance as good as the optimal user. However, although not optimal, AbraQ does not require the user to make any selection and there is no guarantee that the user would make optimal selections. In fact, research [15] has found that from the perspective of recall enhancement, while user selections improve performance, they failed to reach the optimal performance and in general performed worse than even automatic query expansion techniques, which help enhance recall. Similar results are expected for precision enhancement and therefore this has a negative effect on the results shown for all the interactive query refinements methods, which includes Optimal AbraQ, but excludes AbraQ.

Clustering methods help when the query is ambiguous and the different possible meanings of the query have very distinct sets of associated vocabulary. Web log analysis helps when the query is short and other users have frequently refined this query. Relevance feedback helps when the initially retrieved documents are reasonably relevant. Our new method, AbraQ, works independently of the distinctness of the concepts in the relevant and irrelevant documents, the length or frequency of the query, and even works when there are no relevant documents amongst those initially retrieved.

As earlier, significance was tested by comparing each pair of methods using the Wilcoxon signed-rank test. On the first five results, Optimal AbraQ was significantly better than all other methods at a 99% level of confidence. AbraQ, Lingo, and QDC were significantly better than both Google and Mamma at a 95% level of confidence. On the first ten results, Optimal AbraQ was significantly better than all but Lingo and QDC at a 95% level of confidence. AbraQ was significantly better than Google, STC, and Mamma at a 95% level of confidence. There was no significant improvement by any other method.

5.6 Algorithm Complexity and Efficiency

Although most computations performed in the algorithm are relatively cheap, two components are quite costly: counting the number of matching documents (counts), and querying to find matching documents (queries). As the number of aspects grows, so does the amount of counting and querying required. Fortunately, most queries have few words and even fewer aspects: 86% of queries have less than five words, and only 0.4% have more than nine [13].

For a query with n words, and a aspects, identifying the query aspects involves counting the number of matching documents for all permutations of each potential aspect, of which there are $O(n)$. Identifying the underrepresented aspects involves running $O(a^2)$ queries, one for each aspect and one for each pair of aspects. This step also involves computing the co-occurrence strength for $O(a)$ terms, which each require one count. During this step, the algorithm may choose to back off an aspect, this can occur at most $n - a$ times and requires rerunning this step, but fewer queries and counts are required as many will already have been computed. Identifying the refinement involves $O(1)$ queries, one for each refinement tested.

Aspects are typically one to three words in length. Assuming that the maximum length of any aspect is four words, then the worst case occurs when there is one aspect for every word in the query. In this case the number of counts and number of queries performed are given by:

$$Counts = 403n - 3$$

$$Queries = n^2/2 + n/2 + 50$$

That means for queries of 2 to 10 words, between 803 and 4027 counts are required and between 53 and 105 queries are

required. For approximately 10% of aspects, the algorithm will back-off an aspect in the second stage of the algorithm. This incurs less than 20% additional cost in the case with 10 words, or more specifically it incurs an additional 800 counts and $2a - 1$ queries, where a is the number of aspects before the back-off.

Many queries are easy, about 70% contain just one aspect and AbraQ terminates after the first stage, using just a handful of inexpensive counts. Of the remaining queries, about 50% are answered sufficiently by search engines and AbraQ can terminate after the second stage. In these cases, AbraQ's cost is negligible for the majority of queries, as they tend to be short. Most of the cost of AbraQ is in the final stage, finding the refinements, and this only occurs on the remaining queries (15%) that AbraQ can potentially improve significantly. These 15% of queries are amongst the ones that users spend most (over 90%) of their total search time refining [13]. Our analysis has not taken into account the number of queries saved by avoiding the user's refinement process. Future work could analyze the number of refinements and queries saved by AbraQ.

The impact on search responsiveness is small. Queries and counts can be performed in parallel, so AbraQ enhanced queries take no longer from the user's perspective. But additional hardware will be required to process the same number of queries. The most significant component of the cost is the additional queries. When weighted by the frequency of occurrence [13], the average AbraQ refinement will involve 56 queries. Therefore, averaged over all searches, including those where AbraQ is not applicable, AbraQ will increase the number of queries performed by a factor of ten.

Counts are fairly cheap compared to queries. For most, finding the count requires only a lookup in the existing document frequency table of the search engine. For those that much be computed, an approximation is sufficient and this is far cheaper than determining the exact number. Approximations are sufficient, as shown by the experiments in this paper, which used the approximations returned by Google for the counts. Furthermore, the counts are correlated, which further reduces the cost, for instance, the count of the intersection between terms can be computed simultaneously with the count of the individual terms.

The queries can also be computed for significantly less cost than is initially apparent. As with the counts, many queries are correlated, which reduces the cost, for instance, the pairs of aspects in the second stage are correlated with the individual aspects, and the refinement candidates in the third stage are correlated with the user's original query. Additionally, the quality of the matching documents is less important, the only requirement is that the vocabulary is similar. This means that far fewer documents could be ranked than during a normal search, significantly reducing the cost of queries. If approximately one tenth the documents were ranked, this would mean AbraQ would only increase the total number of queries performed by a factor of two. Which is very reasonable, considering the significant benefit for the most time consuming queries.

6. CONCLUSION

This paper has presented AbraQ, a new automatic query expansion algorithm that significantly improves web search performance on hard multi-aspect queries. AbraQ works independently of document characteristics, the length or frequency of the query, and is even independent of there being any relevant documents amongst those initially retrieved. Firstly, AbraQ identifies aspects in an unstructured query, giving a new solution to the query splitting problem. Secondly, AbraQ introduces a way of identifying underrepresented query aspects. Identifying underrepresented aspects is of great significance since the main cause of search failure for a broad range of web search and information retrieval systems is underrepresented aspects in multi-aspect queries. Thirdly, AbraQ is able to automatically select a refinement that addresses underrepresented query aspects.

The evaluation of AbraQ has shown that while other automatic refinement techniques focus on recall with no positive effect on precision, AbraQ significantly improves precision on hard multi-aspect queries. The evaluation also showed that AbraQ successfully identified the aspects and successfully identified whether the aspects were sufficiently represented in the result set. By only applying refinements to hard queries where performance was poor on the underlying search system, the system provides a potential mechanism for estimating query difficulty, which has applications to other areas.

AbraQ was also compared against interactive query refinement methods that leverage additional user input. Even with optimal user input for the other approaches, AbraQ performed at a similar level to the best approaches and the only method to do significantly better was an interactive adaption of AbraQ. Under normal conditions, the other approaches would not achieve optimal performance because of sub-optimal user input, which other researchers have found to be typical, whereas AbraQ would continue performing well.

While the results and contributions of this paper are significant, there is still much room for improvement. Many parts of AbraQ have simplistic implementations and it may be possible to improve the performance of AbraQ significantly by using some of the suggestions identified in the prototype implementation sections of the algorithm, for example, by using Bayesian networks instead of Naive Bayes when identifying underrepresented aspects. Further investigation is needed into the applicability of AbraQ for query difficulty estimation.

7. ACKNOWLEDGMENTS

Daniel Crabtree is supported by a Top Achiever Doctoral Scholarship from the Tertiary Education Commission of New Zealand.

8. REFERENCES

[1] P. Berkhin. Survey of clustering data mining techniques. Technical report, Accrue Software, San Jose, CA, 2002.

[2] C. Buckley. Why current ir engines fail. In *ACM SIGIR*, pages 584–585, New York, NY, USA, 2004. ACM Press.

[3] C. Buckley, G. Salton, J. Allan, and A. Singhal. Automatic query expansion using smart: Trec 3, 1994.

[4] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? In *ACM SIGIR*, pages 390–397, New York, NY, USA, 2006. ACM Press.

[5] R. L. Cilibrasi and P. M. Vitanyi. The google similarity distance. *IEEE Transactions on Knowledge*

and Data Engineering, 19(3):370–383, March 2007.

[6] K. Collins-Thompson and J. Callan. Query expansion using random walk models. In *CIKM*, pages 704–711, 2005.

[7] D. Crabtree, P. Andreae, and X. Gao. Query directed web page clustering. In *Web Intelligence*, pages 202–210, 2006.

[8] B. B. Croft, H. R. Turtle, and D. D. Lewis. The use of phrases and structured queries in information retrieval. In *ACM SIGIR*, pages 32–45. ACM Press, 1991.

[9] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *WWW*, pages 325–332, 2002.

[10] C. de Loupy and P. Bellot. Evaluation of document retrieval systems and query difficulty. In *Using Evaluation within HLT Programs : Results and Trends*, pages 34–40, 2000.

[11] D. Harman. *Relevance feedback and other query modification techniques*, chapter 11, pages 241–263. Englewood Cliffs: Prentice Hall, 1992.

[12] M.-H. Hsu, M.-F. Tsai, and H.-H. Chen. Query expansion with conceptnet and wordnet: An intrinsic comparison. In *AIRS*, pages 1–13, 2006.

[13] B. J. Jansen, A. Spink, and J. O. Pedersen. A temporal comparison of altavista web searching. *JASIST*, 56(6):559–570, 2005.

[14] C. S. G. Khoo and D. C. C. Poo. An expert system approach to online catalog subject searching. *Information Processing and Management*, 30(2):223–238, 1994.

[15] M. Magennis and C. J. van Rijsbergen. The potential and actual effectiveness of interactive query expansion. In *ACM SIGIR*, pages 324–332, 1997.

[16] Mamma.com: www.mamma.com, 2007.

[17] F. A. D. Neves, E. A. Fox, and X. Yu. Connecting topics in document collections with stepping stones and pathways. In *CIKM*, pages 91–98, 2005.

[18] S. Osiński, J. Stefanowski, and D. Weiss. Lingo: Search results clustering algorithm based on singular value decomposition. In *Intelligent Information Processing and Web Mining Conference*, Advances in Soft Computing, pages 359–368, Zakopane, Poland, 2004. Springer.

[19] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976.

[20] J. Rocchio. Relevance feedback in information retrieval. In G. Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. Prentice Hall, 1971.

[21] I. Ruthven and M. Lalmas. A survey on the use of relevance feedback for information access systems. *The Knowledge Engineering Review*, 19(2):95–145, June 2003.

[22] I. Soboroff. On evaluating web search with very few relevant documents. In *SIGIR*, pages 530–531, 2004.

[23] C. J. van Rijsbergen. *Information retrieval*. Butterworths, 2nd edition edition, 1979.

[24] B. Vélez, R. Weiss, M. A. Sheldon, and D. K. Gifford. Fast and effective query refinement. In *ACM SIGIR*, pages 6–15, 1997.

[25] V. Vinay, K. R. Wood, N. Milic-Frayling, and I. J. Cox. Comparing relevance feedback algorithms for web search. In *WWW*, pages 1052–1053, 2005.

[26] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *ACM SIGIR*, pages 4–11, 1996.

[27] E. Yom-Tov, S. Fine, D. Carmel, and A. Darlow. Learning to estimate query difficulty: including applications to missing content detection and distributed information retrieval. In *ACM SIGIR*, pages 512–519, 2005.

[28] X. Yu, F. A. D. Neves, and E. A. Fox. Hard queries can be addressed with query splitting plus stepping stones and pathways. *IEEE Data Engineering Bulletin*, 28(4):29–38, 2005.

[29] O. Zamir and O. Etzioni. Web document clustering: A feasibility demonstration. In *Research and Development in Information Retrieval*, pages 46–54, 1998.

[30] J. Zhang, L. Sun, Y. Lv, and W. Zhang. Relevance feedback by exploring the different feedback source and collection structure. In *Text REtrieval Conference (TREC)*, 2005.