Web Search Intent Induction via Automatic Query Reformulation

Hal Daumé III
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey, CA 90292
hdaume@isi.edu

Eric Brill
Microsoft Research
One Microsoft Way
Seattle, WA 98052
brill@microsoft.com

Abstract

We present a computationally efficient method for automatic grouping of web search results based on reformulating the original query to alternative queries the user may have intended. The method requires no data other than query logs and the standard inverted indices used by most search engines. Our method outperforms standard web search in the task of enabling users to quickly find relevant documents for informational queries.

1 Introduction and Motivation

In a study of web search query logs, Broder (2002) observed that most queries fall into one of three basic categories: *navigational*, *informational* and *transactional*. A navigational query is one where the user has a particular URL they are attempting to find. An informational query is one where the user has a particular information need to satisfy. A transactional query is one in which the user seeks to perform some sort of web-mediated activity (such as purchasing a product).

In that paper, Broder (2002) also confirms that most queries are very short: on the order of two words. For informational queries this is often inadequate. The brevity of queries is often due to the fact that the user does not know exactly what he is looking for. This makes it difficult for him to formulate enough, or even correct, keywords. These types of queries make up anywhere between 39 and 48 percent of all web queries, according to Broder, making them a prime target for research.

2 Prior Work

Our interest is in informational queries. The general approach we explore to assist users find what they want is to present structured results. Dumais et. al. (2001) have shown that displaying structured results improves a user's ability to find relevant documents quickly.

There are three general techniques for presenting web search results in a structured manner, ranging from totally supervised methods to totally unsupervised methods. The first approach, *manual classification*, is typified by a system like Yahoo!, where humans have created a hierarchical structure describing the web and manually classify web pages into this hierarchy. The second approach, *automatic classification* (see, for instance, the classification system reported by Dumais (2000)) builds on the hierarchies constructed for manual classification systems, but web pages are categorized by a (machine-learned) text classification system. The third approach, typified by systems such as Vivisimo and the system of Zamir et al. (1999), look at the text of the returned documents and perform document *clustering*.

A related unsupervised approach to this problem is from Beeferman and Berger (2000). Their approach leverages click-through data to cluster related queries. The intuition behind their method is that if two different queries lead to users clicking on the same URL, then these queries are related (and vice-versa). They perform agglomerative clustering to group queries, based on click-through data.

Our approach is most closely related to this agglomerative clustering approach, but does not require click-through data. Moreover, the use of click-through data can result in query clusters with low user utility (see Section 3.2). Furthermore, our approach does not suffer from the computation cost of document clustering by text and produces structured results with meaningful names without the economic cost of building hierarchies.

3 Methodology

Our goal is to provide a range of possible needs to a user whose query is underspecified. Suppose a naive user John enters a query for "fly fishing." This query will retrieve a large set of documents. We assume that John's search need (information about flies for catching trout) is somewhere in or near this set, but we do not

know exactly where. However, we can attempt to identify *other* queries, made by other people, that are relevant to John's need. We refer to this process as *Query Driven Search Expansion* and henceforth refer to our system as the QDSE system.

3.1 Formal Specification

Formally, if Q is the set of queries to our search engine and D is the set of indexed documents, let R be a binary relation on $Q \times D$ where qRd if and only if d is in the return set for the query q. It is likely that the set of related queries is quite large for a given q (in practice the size is on the order of ten thousand; for our dataset, "fly fishing" has $29,698$ related queries). However, some of these queries will be only tangentially related to q. Moreover, some of them will be very similar to each other. In order to measure these similarities, we define a distance metric between two queries q and q' based on their returned document sets, ignoring the text of the query:

$$\|q, q'\| = 1 - \frac{|R[q] \cap R[q']|}{|R[q] \cup R[q']|} \quad (1)$$

One could then sort the set of related queries according to $\|q, q'\|$ and present the top few to the user. Unfortunately, this is insufficient: the top few are often too similar to each other to provide any new useful information. To get around this problem, we use the maximal marginal relevance (MMR) scheme originally introduced by Carbonell et. al. (1998). In doing so, we order alternative quereies according to:

$$\operatorname*{argmin}_{q'} \left[\lambda \|q, q'\| - (1-\lambda) \min_{q''} \|q', q''\| \right] \quad (2)$$

where q's are drawn from unreturned query expansions and q''s are drawn from the previously returned set.[1]

3.2 Alternative Distance Metrics

One particular thing to note in Equation 1 is that we do not take relative rankings into account in calculating distance. One could define a distance metric weighted by each document's position in the return list.

We ran experiments using PageRank to weight the distance (calculated based on a recent full web crawl). System output was observed to be significantly inferior to the standard ranking. We attribute this degradation to the following: if two queries agree only on their top documents, they are too similar to be worth presenting to the user as alternatives. This is the same weakness as is found in the Beeferman and Berger (2000) approach.

4 System

The system described above functions in a completely automatic fashion and responds in real-time to users queries. Across the top of the return results, the query is listed, as are the top ranked alternative queries. Each of these query suggestions is a link to a heading, which are shown below. Below this list are the top five search result links from MSN Search under the original query[2]. After the top five results from MSN Search, we display each header with a +/− toggle to expand or collapse it. Under each expanded query we list its top 4 results.

5 Evaluation Setup

Evaluating the results of search engine algorithms without embedding these algorithms in an on-line system is a challenge. We evaluate our system against a standard web search algorithm (in our case, MSN Search). Ideally, since our system is focused on informational queries, we would like a corpus of $\langle$query, intent$\rangle$ pairs, where the query is underspecified. One approach would be to create this corpus ourselves. However, doing so would bias the results. An alternative would be to use query logs; unfortunately, these do not include intents. In the next section, we explain how we create such pairs.

5.1 Deriving Query/Intent Pairs

We have a small collection of click-through data, based on experiments run at Microsoft Research over the past year. Given this data, for a particular user and query, we look for the last URL they clicked on and viewed for at least two minutes[3]. We consider all of these documents to be satisfactory solutions for the user's search need. We further discard pairs that were in the top five because we intend to use these pairs to evaluate our system against vanilla MSN Search. Since the first five results our system returns are identical to the first five results MSN Search returns, it is not worthwhile annotating these data-points (this resulted in a removal of about 20% of the data, most of which were navigational queries).

These $\langle$query, URL$\rangle$ pairs give us a hint at how to get to the desired $\langle$query, intent$\rangle$ pairs. For each $\langle$query, URL$\rangle$ pair, we looked at the query itself and the web page at the URL. Given the query, the relevant URL and the top five MSN Search results, we attempted to create a reasonable search intent that was (a) consistent with the query and the URL, but (b) not satisfied by any of the top five results. There were a handful of cases (approximately an additional 5%) where we could not think of a reasonable intent for which (b) held – in these cases, we discarded that pair.[4] In all, we created 52 such pairs; four randomly

[1] Queries that appear to be URLs, and strings with a very small edit distance to the original are discarded.

[2] The top five queries originally returned by MSN Search are included because there is a chance the user knew what he was doing and actually entered a good query.

[3] It may be the case that the users found an earlier URL also to be relevant. This does not concern us, as we do not actually use these URLs for evaluation purposes – we simply use them to gain insight into intents.

[4] We make no claim that the intents we derive were necessarily the original intent in the mind of the user. We only go through this process to get a sense of the sorts of information

chosen ⟨query, URL, intent⟩ triples are shown in Table 1. Once the intents have been derived, the original URLs are thrown away: they are not used in any of our experiments.

5.2 Relevance Annotation

Our evaluation now consists of giving human annotators ⟨query, intent⟩ pairs and having them mark the first relevant URL in the return set (if there is one). However, in order to draw an unbiased comparison between our system and vanilla MSN Search, we need to present the output from both as a simple ordered list. This requires first converting our system's output to a list.

5.2.1 Linearization of QDSE Output

We wish to linearize our results in such a way that the position of the first relevant URL enables us to draw meaningful inferences. In vanilla MSN search, we can ascribe a cost of 1 to reading each URL in the list: having a relevant URL as the 8th position results in a cost of 8.

Similarly, we wish to ascribe a cost to each item in our results. We do this by making the assumption that the user is able to guess (with 100% accuracy) which subcategory a relevant URL will be in (we will evaluate this assumption later). Given this assumption, we say that the cost of a link in the top 5 vanilla MSN links is simply its position on the page. Further down, we assume there is a cost for reading each of the MSN links, as well as a cost for reading each header until you get to the one you want. Finally, there is a cost for reading down the list of links under that header. Given this cost model, we can linearize our results by simply sorting them by cost (in this model, several links will have the same cost – in this case, we fall back to the original ordering).

5.2.2 Annotation

We divided the 52 ⟨query, intent⟩ pairs into two sets of 32 (12 common pairs). Each set of 32 was then scrambled and half were assigned to class *System 1* and half were assigned to class *System 2*. It was ensured that the 12 overlapping pairs were evenly distributed.

Four annotators were selected. The first two were presented with the first 32 pairs and the second two were presented with the second 32 pairs, but with the systems swapped.[5] Annotators were given a query, the intent, and the top 100 documents returned from the search according to the corresponding system (in the case of QDSE, enough alternate queries were selected so that there would be exactly 100 total documents listed). The annotator selected the first link which answered the intent. If there was no relevant link, they recorded that.

people *really* are looking for, so that we need not invent queries off the tops of our heads.

[5]The interface used for evaluation converted the QDSE results into a linear list using our linearization technique so that the interface was consistent for both systems.

5.3 Predictivity Annotation

Our cost function for the linearization of the hierarchical results (see Section 5.2.1) assumes that users are able to predict which category will contain a relevant link. In order to evaluate this assumption, we took our 52 queries and the automatically generated category names for each using the QDSE system. We then presented four new annotators with the queries, intents and categories. They selected the first category which they thought would contain a relevant link. They also were able to select a "None" category if they did not think any would contain relevant links. Each of the four annotators performed exactly the same annotation – it was done four times so agreement could be calculated.

6 Results and Analysis

Our results are calculated on two metrics: relevance and predictivity, as described in the previous section.

6.1 Relevance Results

The results of the evaluation are summarized in Table 2. The table reports four statistics for each of the systems compared. In the table, **MSN** is vanilla MSN search and **QDSE** is the system described in this paper.

The first row is probability of success using this system (number of successful searches divided by the number of total searches). The second line is the probability of success, given that you are only allowed to read the first 20 results. Next, Avg. Success Cost, is the average cost of the relevant URL for that system. This cost averages only over the successes (queries for which a relevant URL was found). The next statistic, Avg. Cost, is the average cost including failures, where the cost of a failure is, in the case of vanilla MSN, the number of returned results and, in the case of QDSE, the cost of reading the top five results, all the labels and one category expansion[6] The last statistic, Avg. Mutual Cost, is the average cost for all pairs where both systems found a relevant document. The last line reports inter-annotator agreement as calculated over the 12 pairs, which is low due partly to the small sample size and partly to the fact that the intents themselves were still somewhat underspecified.[7]

6.2 Predictivity Results

We performed two calculations on the results of the predictivity annotations. In the first calculation, we consider the relevance judgments on the QDSE system to be the gold standard. We calculated accuracy of choosing the correct first category. This measures the extent to which

[6]The user may have been able to determine his search had failed having only read the categories, yielding a lower cost.

[7]We intend to run timed user studies in our future work; however, it has been observed (Dumais et al., 2001) that presenting users with structured results enables them to find relevant documents more quickly; to do timed studies in the linearization is an unrealistic scenario, since one would never deploy the system in this configuration.

Query: Soldering iron	**URL:** www.siliconsolar.com/accessories.htm
Intent: looking for accessories for soldering irons (but not soldering irons themselves)	
Query: Whole Foods	**URL:** www.wholefoodsmarket.com/company/communitygiving.html
Intent: looking for the Whole Foods Market's community giving policy	
Query: final fantasy	**URL:** www.playonline.com/ff11/home/
Intent: looking for a webforum for final fantasy games	
Query: online computer course	**URL:** www.microsoft.com/traincert/
Intent: looking for information on Microsoft Certified Technical Education centers	

Table 1: Four random $\langle$query, URL, intent$\rangle$ triples

	MSN	QDSE
Prob. Success	88.0%	67.7%
Prob. Success 20	68.7%	62.6%
Avg. Success Cost	12.4	4.7
Avg. Cost	22.9	9.0
Avg. Mutual Cost	23.0	9.0
kappa	0.57	0.45

Table 2: Results of the evaluation

the oracle system is correct. On this task, accuracy was 0.54. The second calculation we made was to determine whether a user can predict, looking at the headers only, whether their search has been successful. In the task of simply identifying failed searches, accuracy was 0.70. Inter-annotator agreement for predictivity was somewhat low, with a kappa value of only 0.49.

6.3 Analysis

As can be seen from Table 2, a user is less likely to find a relevant query in the top 100 documents using the QDSE system than using the MSN system. However, this is an artificial task: very few users will actually read through the top 100 returned documents before giving up. At a cutoff of 20 documents, the user is still more likely to succeed using MSN, but the difference is not nearly so large (note, however, that by cutting off at 20 in the QDSE linearization, the user will typically see only one result from each alternate query, thus heavily relying on the underlying search engine to do a good job). The rest of the numbers (not included for brevity) are consistent at 20.

Moreover, as seen in the evaluation of the predictivity results, users can decide, with 70% accuracy, whether their search has failed having read only the category labels. This is in stark contrast to the vanilla MSN search where they could not know without reading all the results whether their search had succeeded.

If one does not wish to give up on recall at all, we could simply list all the MSN search results immediately after the QDSE results. By doing this, we ensure that the probability of success is at least as high for the QDSE system. We can upper-bound the additional cost this would incur to the QDSE system by 4.15, yielding an upper bound of 13.2, still superior to vanilla MSN.

If one is optimistic and is willing to assume that a user will know based only on the category labels whether or not their search has succeeded, then the relevant comparison from Table 2 is between Avg. Success Cost for QDSE and Avg. Cost for MSN. In this case, our cost of 4.7 is a factor of 5 better than the MSN cost. If, on the other hand, one is pessimistic and believes that a user will not be able to identify based on the category names whether or not their search has succeeded in the QDSE system, then the interesting comparison is between the Avg. Costs for MSN and QDSE. Both favor QDSE.

Lastly, the reciprocal rank statistic at 20 results confirm that the QDSE system is more able to direct the user to relevant documents than vanilla MSN search.

7 Conclusion

We have presented a method for providing useful suggested queries for underspecified informational queries. We evaluated our system using an unbiased metric against a standard web search system and found that our system enables users to more quickly find relevant pages. This conclusion is based on an "oracle" assumption, which we also evaluate. Based on these evaluations, we can show that even under a pessimistic view point, our system outperforms the vanilla search engine.

There is still room for improvement, especially in the predictivity results. We would like users to be able to more readily identify the class into which a relevant document (if one exists) would be found. We are investigating multi-document summarization techniques which might allow users to better pinpoint the category in which a relevant document might be found.

References

D. Beeferman and A. Berger. 2000. Agglomerative clustering of a search engine query log. In *KDD*.

S. Brin and L Page. 1998. The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems*.

A. Broder. 2002. A taxonomy of web search. In *SIGIR*.

J. Carbonell and J. Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Research and Development in Information Retrieval*.

S. Dumais and H. Chen. 2000. Hierarchical classification of Web content. In *Proc. of SIGIR-00*.

S. Dumais, E. Cutrell, and H. Chen. 2001. Optimizing search by showing results in context. In *CHI*.

O. Zamir and O. Etzioni. 1999. Grouper: a dynamic clustering interface to Web search results. In *Computer Networks*.