Mining Historic Query Trails to Label Long and Rare Search Engine Queries

PETER BAILEY
Microsoft
RYEN W. WHITE
Microsoft Research
HAN LIU
Carnegie Mellon University
and
GIRIDHAR KUMARAN
Microsoft

Web search engines can perform poorly for long queries (i.e., those containing four or more terms), in part because of their high level of query specificity. The automatic assignment of labels to long queries can capture aspects of a user's search intent that may not be apparent from the terms in the query. This affords search result matching or reranking based on queries and labels rather than the query text alone. Query labels can be derived from interaction logs generated from many users' search result clicks or from *query trails* comprising the chain of URLs visited following query submission. However, since long queries are typically rare, they are difficult to label in this way because little or no historic log data exists for them. A subset of these queries may be amenable to labeling by detecting similarities between parts of a long and rare query and the queries which appear in logs. In this article, we present the comparison of four similarity algorithms for the automatic assignment of Open Directory Project category labels to long and rare queries, based solely on matching against similar satisfied query trails extracted from log data. Our findings show that although the similarity-matching algorithms we investigated have tradeoffs in terms of coverage and accuracy, one algorithm that bases similarity on a popular search result ranking function (effectively regarding potentially-similar queries as "documents") outperforms the others. We find that it is possible to correctly predict the top label better than one in five times, even when no past query trail exactly matches the long and rare query. We show that these labels can be used to reorder top-ranked search results leading to a significant improvement in retrieval performance over baselines that do not utilize query labeling, but instead rank results

Authors' addresses: P. Bailey, R. W. White, and G. Kumaran, Microsoft Corporation, One Microsoft Way, Redmond, WA 98052; email: {pbailey, ryenw, giridhar}@microsoft.com; H. Liu, Carnegie Mellon University, School of Computer Science, 5000 Forbes Avenue, Pittsburgh, PA 15213; email: hanliu@cs.cmu.edu.

using content-matching or click-through logs. The outcomes of our research have implications for search providers attempting to provide users with highly-relevant search results for long queries.

Categories and Subject Descriptors: H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*Performance evaluation*

General Terms: Experimentation, Measurement

Additional Key Words and Phrases: Long queries, query labeling

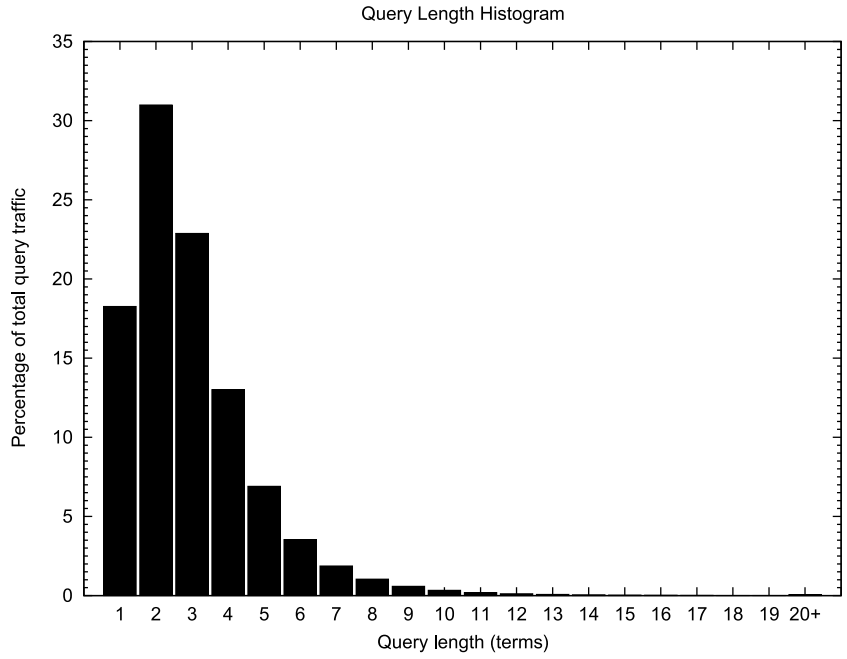
ACM Reference Format:
Bailey, P., White, R. W., Liu, H., and Kumaran, S. 2010. Mining historic query trails to label long and rare search engine queries. ACM Trans. Web, 4, 4, Article 15 (September 2010), 27 pages. DOI = 10.1145/1841909.1841912. http://doi.acm.org/10.1145/1841909.1841912.

1. INTRODUCTION

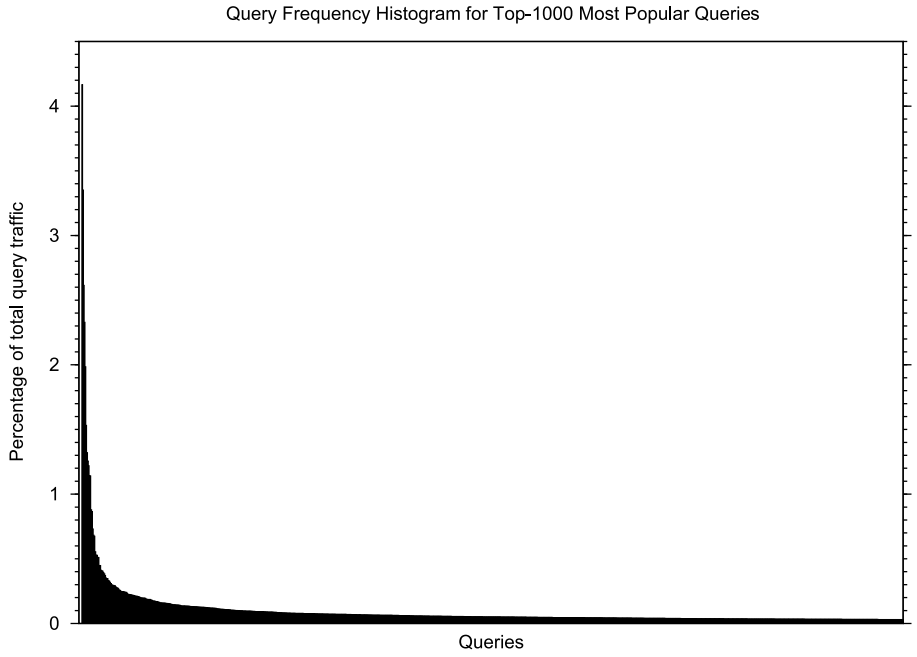
Web search engine users, working with limited support for query specification, may use longer-than-usual query statements to provide additional information about their needs to the engine [Kumaran and Allan 2007]. However, handling such rich representations is difficult for search engines since the query text usually contains a lot of noise, such as extraneous terms that users believe are important to conveying their information needs, but in fact are confusing to automatic systems [Kumaran and Allan 2008]. One way to address this challenge is by associating category labels with queries. For example, the top category labels drawn from the Open Directory Project (ODP, http://www.dmoz.org) for the query [t-bills united states treasury] might be *Business/Investing/Stocks_and_Bonds*, *Society/Government/Finance*, and *Business/Investing/Guides*. These labels may provide valuable additional information to augment retrieval operations (e.g., as additional ranking features or for postretrieval result reranking).

We can assign query labels using interaction log data containing users' postquery interactions [Li et al. 2008]. Search engines such as Google, Yahoo!, and Microsoft's Bing gather extensive interaction log data pertaining to each query via many users' result click-through and postquery browsing. These data can be used for improving search engine performance [Agichtein et al. 2006; Bilenko and White 2008] or for search behavior analysis [White and Drucker 2007]. For queries that occur frequently, interaction data is plentiful. In such circumstances, it is relatively straightforward to assign labels using click-through records describing which search results users visit, a method for labeling those pages, and the aggregation of those labels across multiple queries and/or users.

For the purposes of this research, we define long queries as those containing four or more query terms. The distribution of query frequencies in log data is strongly weighted towards short queries, ranging from one to three terms in length. Figure 1(a) illustrates the typical query length frequency distribution, obtained from a multimonth sample of major Web search engine interaction logs. Combined, long queries make up over a quarter of the volume of queries from which this plot was generated. While it is possible that popular queries could be long as well, in practice they are short. Short queries form the "head" of the query histogram. For queries that occur rarely, the interaction



(a) Length histogram



(b) Frequency histogram

Fig. 1. Query length and frequency histograms.

log data is sparse to nonexistent. These queries form the "tail" of the query histogram. This can be seen visually in the query histogram in Figure 1(b). In fact, the Spearman rank correlation coefficient between query popularity and query length in our sample is -0.28, signifying that query popularity is inversely related to query length. Long queries are generally rare, making them unamenable to log-based query labeling, but they are also important to search engines given the significant proportion of the query volume that they represent.

To assign accurate labels to queries where little or no interaction data exists, we present an approach that leverages queries for which we can assign labels, mines these queries for those that are similar to rare long queries of interest, and propagates the labels from them to the rare long queries. We use a labeling mechanism with a large number of categories drawn from the Open Directory Project[1] (ODP), as we believe that long queries tend to have more specific query intent (an assertion supported by previous work [Phan et al. 2007]), and thus narrower categories will more precisely reflect the query. We employ the postquery interactions of many search engine users (referred to as their *query trails*) to create a labeled set of past queries, and experiment with multiple query similarity functions that allow us to leverage such sets to label rare long queries.

The contribution of this article is twofold. First, we demonstrate that it is possible to accurately predict the category labels for a significant proportion of long and rare queries using historic log data, and draw conclusions about the most effective label prediction algorithms for this task given related work in areas such as probabilistic retrieval, language modeling, and named entity extraction. Second, we show that the category labels assigned to long and rare queries have utility in improving retrieval effectiveness when used to rerank the top results obtained using a state-of-the-art retrieval algorithm.

The remainder of this article is structured as follows. In Section 2 we describe related work in processing long queries and query labeling. Section 3 describes the experimental framework, including the data sets and the algorithms tested. Section 4 presents the evaluation methods, including the metrics used to determine algorithm performance. The findings of our analysis are presented and discussed in Section 5. In Section 6 we demonstrate through experimentation how the assigned labels can be used to improve search result relevance. We conclude in Section 7, including some important opportunities for future research.

2. RELATED WORK

The significant areas of related work pertaining to the research we describe in this article are: prior work on processing long queries and work on query labeling and classification. Information from the ODP has also been used for result reranking activities.

[1]http://www.dmoz.org

2.1 Long Queries

Prior research work on long queries has generally focused on the development of automatic techniques for the reduction of long queries to a shorter form of the same query that may be more effective for retrieval [Bendersky and Croft 2008; Lease et al. 2009; Kumaran and Carvalho 2009]. Techniques include the learning of key concepts in verbose queries using query, collection-dependent, and collection-independent features [Allan et al. 1997; Bendersky and Croft 2008], the conversion of verbose queries into structured queries [Callan et al. 1995], reweighting all terms in long queries [Lease et al. 2009], or the selection of high-quality subqueries by predicting the quality (or performance) of those subqueries [Kumaran and Carvalho 2009]. Kumaran and Allan [2008] addresses the issue of extracting the optimal subquery (in terms of retrieval effectiveness) from the original long query. Their approach involves extracting a short list of candidate subqueries using a mutual information measure and presenting this list to the user, allowing her to replace the original query by one of the candidates from the list. This approach resulted in significant improvements over retrieval with the original long queries. The reduction of queries to shorter form—either automatically or interactively—resembles some of the research described in this article. However, we are focused on identifying similar, not necessarily reduced, queries. We also use interaction logs rather than machine learning or natural language processing techniques. Lastly our task is to assign labels to queries to create a richer representation of information needs that can be used for a range of purposes, of which more effective information retrieval (IR) is only one example, rather than finding the best subquery that maximizes retrieval performance.

2.2 Query Labeling and Classification

Relevant related work on query labeling and classification can be organized into three main groups: (i) Web-based methods, where Web search results for the query are retrieved and then classified; (ii) interaction-log-based methods, where the search queries and other Web interactions of users are logged and mined for patterns of behavior, and; (iii) term matching and machine-learning methods, where queries are matched to other queries by features. The selection of previous work we present in this subsection is representative, but not exhaustive.

The work of Broder et al. [2007] is an example of a Web-based method for query classification. In a similar way to our research reported in this article, they focus on classifying rare queries using thousands of categories from a commercial taxonomy. Query labels were determined by classifying the corresponding retrieved Web documents for each query and applying these classifications to the queries. They achieved a classification performance significantly better than a nearest-neighbor-based baseline algorithm. The main assumption for such Web-based methods is that the top retrieved Web documents are highly relevant to the query. The main purpose of their work was to use these query labels for better matching of online advertisements with rare queries.

Another Web-based approach is proposed by Shen et al. [2005] for their participation for the KDD Cup 2005 [Li et al. 2005]. In their approach, Web documents from multiple major search engines are first classified into the ODP hierarchy using an ensemble of classifiers. These classification results were then mapped to the target taxonomy for the query classification. Other Web-based methods include Kardkovács et al. [2005] and Vogel et al. [2005]. As discussed previously, these Web-based methods make the assumption that the search engines are reasonably good at returning relevant documents for the given query, such that the top retrieved results are highly correlated with the query intent. However, this solution may not be practical for long and rare queries since Web search engines typically use a logical AND operator in their underlying retrieval models. This means the result set may be small or contain no results at all, making classification challenging or impossible.

Related work on the use of interaction logs for query classification was conducted by Li et al. [2008]. They report on leveraging click graphs obtained from interaction logs to drastically increase the amount of training data classification methods receive. Using their method, a small set of *seed queries* are first manually labeled. The label information then propagates through the click graph to the unlabeled queries until a global equilibrium state is achieved. By applying some regularization techniques, they prevent a click graph from propagating erroneous labels. The success for such a graph-based semi-parametric learning approach depends on the selection and quality of the small set of seed queries. If the number of query labels is small, this is easier to do. However, if the number of distinct query labels extends into many thousands or beyond, it is not obvious how this small set of seed queries can be correctly selected. Similar work by Beitzel et al. [2005] leveraged unlabeled data to improve the supervised learning performance.

Previous work has also investigated term matching methods for query labeling. The simplest idea to label a query is by looking it up in a database of manually classified queries, as discussed by Beitzel et al. [2007]. A drawback with such an exact matching approach is that queries are structurally diverse. Punctuation and search operators may be misused or misparsed, and keywords can be altered by synonyms with similar meanings. This can result in exact term matching occurring infrequently. The authors also propose a relaxed n-gram matching strategy to mitigate this problem. For n-grams, a query is partitioned into a set of substrings that are n units long, where units can be words or characters. From their results, even though the n-gram matching substantially increases recall over exact matching, it is limited for our purpose by the difficulties in covering the rare queries in the long tail. Similar techniques include query expansion through external dictionaries, for example, Voorhees [1994]; query refinement through explicit interactive feedback [Allan and Raghavan 2002]; and query enhancement through implicit feedback [Gravano et al. 2003]. However, dictionary-based solutions may encounter coverage issues, especially for long and rare queries. As already stated in the previous section, we required a solution that was noninteractive to work at scale and reduce user burden, so any method that requires user feedback (explicit or implicit) was inappropriate for our purposes.

2.3 Use of ODP for Result Reranking

We use ODP information to rerank search results in a demonstration of how query labels can be used to benefit users. Bennett and colleagues assigned ODP category labels to top-ranked search results and showed that these labels could boost retrieval performance [Bennett et al. 2010]. Chirita et al. [2005] used the topics contained within the ODP as a way for users to characterize their interest profiles. Web results are then reranked according to their distance from the user's profile. They found an increase in relevance performance of Google results reranked according to a personalized algorithm that promoted Web results that had been classified within these ODP groups. Qiu and Cho [2006] build on this work by providing a mechanism whereby a user's topic-sensitive profile can be automatically inferred from their click patterns.

In the next section we describe the experimental framework used for our study. This includes a description of the data sets and the four similarity matching algorithms used to find queries related to a given long and rare query.

3. EXPERIMENTAL FRAMEWORK

The primary aim of our study was to investigate the performance of different label prediction algorithms for long and rare queries. To achieve this, we required a query labeling algorithm, the output of which we could consider as the ground truth for our experiments.

To develop this ground truth we used a rich set of interaction logs containing the querying and browsing behavior of around 500,000 users of the widely distributed Windows Live toolbar during 2008. Toolbar users consented to having their interaction logged, and all user-identifying information was stripped from the logs prior to use in this study. The information contained in these log entries included a unique identifier for the toolbar, a timestamp for each page view, a unique browser window identifier, and the URL of the Web page visited. Intranet and secure (https) URL visits were excluded at the source. In order to remove variability caused by geographic and linguistic variation in search behavior, we include only entries generated in the English-speaking United States locale.

From these logs we obtained information about users' Web interaction behavior, including the search queries they issue to all popular engines, the Web pages they visit, and their dwell time on each of the pages. These logs allowed us to utilize all Web searching behavior rather than the subset that is a visible from a particular search engine.

To assign labels to queries we extracted *query trails* [Bilenko and White 2008] from the logs. A query trail qt comprises a user's query q (consisting of a sequence of terms $\{t_1, t_2, ...t_{|q|}\}$), and a sequence of zero or more visited documents identified by URL $\{u_1, u_2, ..., u_n\}$, visited on the click trail following q. (For easy reference, all notation used throughout the article is summarized in Table I).

We considered only *satisfied* query trails *sqt*: these are trails where there is no signal of the user abandoning the query. Abandonment signals include: no URLs being clicked in response to the query, no URLs being visited for more

Table I. Notation Used for Describing the Framework and Algorithms

q	query
eq	an equivalence query (i.e., query same as another, perhaps through normalization)
t	query term
ne	named entity (from Wikipedia)
u	a URL
d	a word within a document (URL)
l	ODP category label for a URL
c	frequency count of a category label's occurrence
pl	a predicted ODP category label
qt	query trail, comprising a query and subsequent visited URLs
sqt	satisfied query trail, with no sign of user dissatisfaction or abandonment (e.g., no short dwell time)
su	satisfying URL in a sqt (i.e., page view with no dissatisfaction or abandonment signal)
lqt	labeled query trail, with URLs labeled using ODP categories
$lqt.query$	the query from a labeled query trail
$alqt$	aggregated labeled query trails, comprising ODP labels and their frequency counts from all query trails with equivalent queries
$alqt.query$	the equivalence query from an aggregated labeled query trail
$trails_h$	historical trails, all query trails from six months of log data (used in training)
$trails_{he}$	historical trails, queries exactly as issued
$trails_{hn}$	historical trails, queries normalized
$trails_t$	test trails, from two months of log data (used in testing)
$trails_{te}$	test trails, queries exactly as issued
$trails_{tn}$	test trails, queries normalized

than five seconds (based on empirical evidence), and no transition to another search engine. An sqt consists of a user's query q and a sequence of one or more visited Web pages that meet our criteria for indicating user satisfaction and are identified by URL $\{su_1, su_2, ..., su_s\}$.

The labeler automatically assigned leaf-level category labels from the ODP taxonomy to Web pages in sqt by performing directory lookup on the page URL in a manner similar to Shen et al. [2005]. These labels were then associated with the query that initiated the trail. The basic method was to assign labels using a backoff strategy, such that if su_i fails to match any of the URLs in the ODP, successive URL path fragments are removed from the end of su_i and the residual URL retried for a match within the ODP. Note that su_i (or some path fragment thereof) must exist within ODP for it to be labeled, and that it is labeled with the full ODP category (e.g., *Recreation/Outdoors/Hiking/Books_and_Maps*) rather than, say, the first-level category (e.g., *Recreation*). Exactly one ODP category is reported by the labeler, although identical URLs may appear in multiple categories. However, since our labeler is deterministic, it is unimportant for any individual URL as to which category it is labeled with.

A labeled query trail lqt consists of a user's query q (comprising a sequence of terms $\{t_1, t_2, ..., t_{|q|}\}$ and sometimes written as $lqt.query$), and a sequence of one or more tuples consisting of ODP category labels and counts $\{(l_1, c_1), (l_2, c_2), ..., (l_m, c_m)\}$. A count c_j is determined by the number of URLs su_i which were labeled with any specific category label l_j. For example, an lqt

generated for a user searching for information on backpacking might resemble the following.

Initial query: [ultralight backpacking]

Recreation/*Outdoors*/*Hiking*/*Backpacking*/*Ultralight_Backpacking* : 7

Recreation/*Outdoors*/*Hiking*/*Organizations*/*North_America* : 4

Recreation/*Outdoors*/*Hiking*/*Trails* : 2

Note that the number of category labels in *lqt* may be substantially fewer than the number of URLs in the corresponding satisfied query trail *sqt*, since multiple URLs (e.g., $\{u_i, u_k, \ldots\}$) may be assigned the same category label l_j. Note also that an *sqt* may have no corresponding *lqt*, if none of the URLs in *sqt* can be labeled.

For the purposes of our experiment, we further transformed the labeled query trail data to create aggregated labeled query trails (*alqt*), where aggregation was by equivalence of their component query q. The way in which equivalence between queries is determined forms one aspect of the data preparation that may significantly affect results, as discussed in Section 3.2. An *alqt* consists of an equivalence query *eq* (consisting of a sequence of terms $\{t_1, t_2, \ldots, t_{|q|}\}$), and a sequence of one or more tuples (l_i, c_i) such that l_i appears if it appears in any of the *lqt* that were aggregated, and such that the count c_i for a label l_i is the sum of the counts for l_i in each of the individual *lqt* which were aggregated.

3.1 Datasets

To investigate labeling previously unseen queries, we split our aggregated labeled query trails into two time periods: *historical* and *test*. The historical data ($trails_h$) consisted of all aggregated labeled query trails from six months of toolbar logs. Our particular interest is in long and rare queries, where rarity is to the point of being unseen with respect to past query trails. We thus chose test data ($trails_t$) consisting of aggregated labeled query trails from the succeeding two months' worth of toolbar logs, such that: (1) equivalence queries were more than three terms in length; and (2) no occurrence of the equivalence query *eq* existed in $trails_h$. All trails whose equivalence queries did not meet these two criteria were discarded from the test data set.

Our test data contained entirely queries which emulate the real-world challenge of having never been seen prior to their issuance in a search engine. Indeed, 30.4% of the unique queries in the two-month period were not issued to any engine in the previous six months (illustrating the importance of effectively handling rare queries) and 16.8% of unique queries were both long and unseen (illustrating the importance of effectively handling long and rare queries). Our goal is to accurately predict the labels for these long and unseen queries using search trail information. We develop different algorithms for this purpose, which use labels assigned to similar seen queries for which we do have log data. To evaluate these algorithms we develop a ground truth set for $trails_t$, containing queries labeled according to the ODP labeling algorithm. That is, we use the query trails in the unseen set, label them with ODP categories per

the technique described in the previous subsection, and use these labels as the ground truth for evaluating our label predictions based on coverage and relevance. This labeling process allows us to perform automated judging, which is essential given the millions of trails present in our test set. Although used for the construction of the ground truth, no prior usage data about the specific queries being processed is made available to the similarity matching algorithms (i.e., the queries must remain unseen).

3.2 Variations in Data Preparation

Any investigation of long queries is open to pursuing different ways of processing the data, and alternate outcomes in the performance of different labeling methods. One dimension is the treatment of past queries—whether to normalize them or not, and to what extent. Query normalization refers to a set of actions taken to alter queries so that small variations in language or expression are not considered to be significant.

In the following text, we adopt the notation [termA termB] to indicate a query consisting of the terms and punctuation between the pair of brackets []. For example, the query [pictures mount baker] and the query [Mount Baker pictures] are syntactically different. However, if lower-casing and term-sorting are applied, both queries become [baker mount pictures]. This approach follows on our intuition that there are many queries for which the intent is identical, even if the expression of the intent varies. Query normalization can cause false equivalence, however. For example, [man bites dog] and [dog bites man] both normalize to [bites dog man] even though the semantics of the two queries are different.

We prepared two variants of the trails data: one exactly as originally recorded in our interaction logs ($trails_{he}$ and $trails_{te}$), and one with normalization processing of queries ($trails_{hn}$ and $trails_{tn}$). The normalization that we applied consisted of converting all terms to lower case, replacing all sequences of whitespace characters with a single space character, removing all punctuation characters, removing English-language plurals indicated by a trailing *s* character, and sorting in alphabetical order. According to this logic, the query [Mount Rainier's scenic hiking trails] thus becomes [hiking mount rainier scenic trail]. Since our logs were from users in the English-speaking United States ISO locale, we did not observe many non-English queries. Non-English queries that were observed were removed from our analysis. We chose to retain stop words (e.g., "a," "the," "and") in the current analysis since we analyze completely unseen queries and we felt that stop words may be an important aspect of long queries, and hence worth retaining. Future work will examine the effect of removing stop words.

Table II reports on the percentage of labeled query trails and aggregated labeled query trails in each set, relative to the labeled query trails for exact query equivalence. Since we use proprietary search data for our analysis, we are unable to provide exact numbers in this article. However, we can say that we use hundreds of millions of trails in total, and that our test sets contain tens of millions of trails. Note that the number of *lqt* using normalized query

Table II. Percentage Sizes by Dataset and Preparation, Relative to the Labeled Query Trails for Exact Query Equivalence

	Exact		*Normalized*	
	alqt	*lqt*	*alqt*	*lqt*
historical	24.2%	100.0%	13.9%	99.9%
test	3.9%	6.8%	2.3%	4.8%

equivalence is a little less than for exact query equivalence. This arises because we discard *eq* that are empty of terms after normalization, which can happen if the entire query consists of punctuation. However the number of *alqt* almost halves following normalization, indicating many very close repetitions in all queries, short and long. Note also that $trails_{tn}$ and $trails_{te}$ are substantially smaller than $trails_{hn}$ and $trails_{he}$ respectively, because they are extracted from only two months' of data and also because they originate with queries that are both long and unseen. However the size of these test sets is still over 16% of the size of the historical trails sets (i.e., 3.9% versus 24.2% and 2.3% versus 13.9%), which indicates that long and unseen queries constitute a significant percentage of the query load for Web search engines.

3.3 Matching Algorithms

A variety of similarity matching algorithms were employed to perform query-to-query matching of long and unseen queries with previously seen queries present in the logs. The objective was to issue each test query to the similarity matching algorithm (SMA), and for the algorithm to return an ordered set of historical aggregated labeled query trails (*alqt*) that the algorithm determined to be related by query similarity. The general process for an individual mapping of a query to aggregated labeled query trails can be denoted as: $SMA(q) \rightarrow \{alqt^{\prime}, ..., alqt^{n}\}$, such that $alqt^{\prime}, \ldots, alqt^{n} \in trails_h$.

Once a set of similar previous *alqt* had been retrieved from the historic trails data ($trails_h$), the labels from that set could be used to generate query labels for the long and unseen test query. In the remainder of this section we describe each of the four similarity algorithms that we developed and tested in this study.

3.3.1 *Term Dropping (TD).* The first algorithm developed dropped the least frequent term from the unseen query, and then found historical queries which included all of the remaining terms.

Term frequency is determined from the statistics of the historical query set, the same set as used elsewhere in this study. The term dropping algorithm works as follows:

$TD(q) \rightarrow \{alqt^{a}, ..., alqt^{n}\}$, such that for $q = \{t_1, t_2, ..., t_i, ..., t_{|q|}\}$, with t_i having the least frequent term occurrence in the queries of $trails_h$, then $alqt^{a}, ..., alqt^{n} \in trails_h$, and $alqt^{f}.query$ contains terms $\{t_1, t_2, ..., t_{i-1}, t_{i+1}, ..., t^{|q|}\}, \forall f \in 1..n$.

The rationale for this algorithm is that we anticipate the query intent behind more complex queries may be partly satisfied by simpler related queries. For example, dropping the least frequent term from the query [TREK mountain

bikes oclv] would leave [TREK mountain bikes], and other queries containing these three terms would be considered as similar queries. This example demonstrates a likely positive application of the algorithm where the least frequent term modifies the core concept (of mountain bikes manufactured by TREK) and query trails related to that concept might well share similar query intent. Our intuition behind term dropping is supported by previous research described earlier in this article, where long queries are reduced to shorter forms to improve retrieval effectiveness [Bendersky and Croft 2008; Lease et al. 2009; Kumaran and Carvalho 2009].

While term dropping is necessary (to match unseen queries with seen queries for which we have historic log data) and may help find similar queries, term dropping may also lead to poor performance in cases where term frequency statistics do not work so well. For example, dropping the least frequent term from the query [where is the bank in Boston] would leave [where is the bank in]. Queries containing these terms may capture the general intent of locating banks, but not the specific location intent related to Boston, which is likely to be crucial to a good match.

3.3.2 *Named Entities (NE).* We considered ways to match specific semantic components of the query to past queries. A number of methods to identify specific semantic intent within the query are appropriate, including part of speech tagging, named entity recognition, term specificity data, and n-gram frequencies. We ultimately settled on using named entity recognition to identify the candidate concepts within queries which should be treated as a unit, rather than as separate terms. Our approach is a variant of that for named entities suggested by Cucerzan [2007], where the surface forms are restricted to the query under consideration. Using this approach, entities in queries are identified based on whether there is a current entry in Wikipedia[2] for the entity. For example, given the query [freestyle snowboarding on Crystal Mountain], "snowboarding" and "Crystal Mountain" would be identified as named entities present in Wikipedia and used to match against the set of historic *alqt*. Around one million entities were extracted from Wikipedia and used in the named-entity assignment. All past queries that contained at least one named entity in common with one in the test queries were considered similar. Queries were ranked based on the degree of named entity similarity. Briefly:

$NE(q) \rightarrow \{alqt^a, \ldots, alqt^n\}$, such that for $q = \{t_1, \ldots, ne_i, \ldots, ne_k, \ldots t_{|q|}\}$, where $ne_i, \ldots, ne_k$ are named entities from Wikipedia, then $alqt^j.query$ contains one or more $ne_j \in \{ne_i, \ldots, ne_k\}$.

This yielded a list of *alqt*, ranked in descending based on the degree of named-entity match between test query and each *alqt*.

When investigating the role of query normalization, named entities were initially assigned to all unnormalized variants of the normalized query, and then aggregated and de-duplicated to assign the entities to the normalized query.

[2]http://www.wikipedia.org

The query similarity challenge can be regarded as a query ranking problem. In addition to term dropping and named entity recognition, we also investigated two ranking methods: language modeling and BM25.

3.3.3 *Language Modeling (LM).* For language modeling, we used version 2.7 of the Indri information retrieval system [Strohman et al. 2005], and employed the query-likelihood variant of statistical language modeling for our language modeling experiments. The query-likelihood approach models queries as being generated by a random sampling from the probabilistic model of documents [Ponte and Croft 1998]. Given a query $q = \{t_1, t_2, t_3, \ldots, t_n\}$, and a query "document" $D = d_1 d_2 d_3 \ldots d_m$, we estimate $P(q|D)$, the probability that the query would be generated by the query "document." Next, by applying Bayesian inversion, we estimate $P(D|q)$, and use this value to rank all the documents in a collection.

Assuming that the terms in the query are independent of each other, $P(q|D) = \prod_{i=1}^{n} P(t_i|D)$, where $P(t_i|D)$ is estimated as $P(t_i|D) = \frac{c(t_i;D)+\mu c(t_i;C)}{|D|+\mu}$ and where $c(t_i; D)$ represents the number of times term t_i occurs in query "document" D, $c(t_i; C)$ represents the number of times t_i occurs in the entire collection of documents C, and $|D|$ is the document's length. We used Dirichlet smoothing [Zhai and Lafferty 2001], a technique that assigns some probability mass from the background collection model to the probability estimate for each query term. This allowed us to avoid the zero-frequency problem, that is, even if a particular term does not occur in a query "document," it does not lead to the $P(q|D)$ estimate evaluating to zero. The amount of probability mass is determined by the value of the parameter μ. For our experiments, we set μ to five. This parameter value was determined by conducting an extensive set of parameter sweeps for normalized and exact queries and maximizing for best performance. This parameter tuning helped ensure that the algorithm was competitive. Upon studying exact queries we made adjustments to the query pre-processing components of Indri so that punctuation, case, and query term order were preserved.

The language modeling approach ranks the entire set of queries (or at least, all queries with at least one term in common with the test query) from $trails_h$. Assuming $LMScore$, we have:

$$LM(q) \rightarrow \{alqt^a, \ldots, alqt^n\}, \text{ s.t. } LMScore(q, alqt^a.query) \geq$$
$$LMScore(q, alqt^b.query) \ldots \geq LMScore(q, alqt^n.query).$$

This yielded an unmanageably large set of *alqt*, ranked in descending order of similarity per $LMScore$. We chose two retrieval thresholds for n, set top 3 and 10, that limited the number of top-ranked similar queries used in query labeling. A sweep of the n values through each value in the range 1..20 revealed that language modeling performance decreased slightly as the retrieval threshold increased.

3.3.4 *BM25.* The second conventional ranking system we used was an implementation of Okapi BM25 [Robertson et al. 1994]. We implemented a BM25 scoring function ($BM25Score$). The tuning weights, k_1 and b, for $BM25Score$

were set to the conventional values of $k_1 = 2.0$ and $b = 0.75$. As with language modeling, we treated the queries from $trails_h$ as documents when indexing. For our purposes, we have a similarity matching BM25 algorithm which returns a ranked list of *alqt* as follows:

$$BM25(q) \rightarrow \{alqt^a, \ldots, alqt^n\}, \text{ s.t. } BM25Score(q, alqt^a.query) \geq BM25Score(q, alqt^b.query) \ldots \geq BM25Score(q, alqt^n.query).$$

Since BM25 ranks the entire set of queries (or at least, all those with at least one term in common) from $trails_h$, we then chose two retrieval thresholds for n. These thresholds were 3 and 10, matching those for the language modeling approach.

In this section we have described four query similarity methods used in our study. The methods we selected covered a range of possibilities, could be applied easily at scale, and demonstrate the plausibility of query labeling using our methodology. Other similarity methods may also be appropriate, including those tailored to short segments of text [Metzler et al. 2007] or semantic similarity [Bollegala et al. 2007]. Future work should test other similarity matching algorithms using this experimental framework.

3.4 Label Prediction

All matching algorithms produced lists of historical *alqt* whose queries were considered similar to each test query. From these lists, we required category label predictions (in rank order) for each test query.

The next step was, therefore, for every test query, to extract from an *alqt* the corresponding set of category labels and counts associated with each similar past *alqt.query*. A set of sets of category labels was therefore available for each test query. From these, we wished to produce a single ranked list of category labels. Given the varying matching algorithms used (and thus the different combining approaches for each algorithm that would be possible given information about scores and/or ranks of the similar historical queries), we chose a simple approach of combining the sets of categories into a ranked list. By using the same combining algorithm in all cases, we maintained the focus of this research on the underlying performance of the similarity matching algorithms. The approach used was to pool the sets of categories into a single set, scoring each category label by the sum of the corresponding counts from the *alqt* sets it had appeared in. The set was then ordered by this frequency score to form a ranked list. At the end of this step, we had for each test query in $trails_{te}$ and $trails_{tn}$ a ranked lists of predicted category labels (written $\{pl_1, pl_2, \ldots, pl_p\}$) for each similarity algorithm and its ranked list of actual category labels (written $\{l_1, l_2, \ldots, l_n\}$) which served as ground truth. For example, for the query [staining a new deck], the three predicted labels in ranked frequency order might be: {*Business*/*Chemicals*/*Coatings_and_Adhesives*, *Home*/*Homemaking*, *Home*/*Gardening*/*Forums_and_Online_Communities*}, and the three ground truth labels in ranked order might be {*Shopping*/*Home_and_Garden*/*Home_Improvement*, *Business*/*Chemicals*/*Coatings_and_Adhesives*, *Home*/*Homemaking*}. Manual verification was performed on the labeling accuracy of the predictions using a

small set of query trails that were not used in training or testing our predictive models.

In this section we have described the datasets we used, the variations in data preparation to consider issues surrounding query normalization, the similarity matching algorithms to identify queries highly similar to a given long and unseen test query, and the generation of predicted labels based on similar queries. In the next section we describe the evaluation methods used to determine the predictive performance of the matching algorithms for labeling long and unseen queries.

4. EVALUATION METHODS

The evaluation task is to predict actual query labels for a given set of long and unseen test queries based on historic queries and the associated labels generated from their postquery navigation trails. We evaluate the predictive performance of the similarity matching functions based on two important measures: *coverage* and *relevance*. We now describe how we interpret these measures in our study.

4.1 Coverage

Coverage reflects the ability of the matching algorithm to generate predictions for any given *alqt.query* seen in $trails_{te}$ or $trails_{tn}$. Obviously the lower the coverage, the less satisfactory the approach is likely to be for a general solution. However, it is possible that highly effective algorithms might be selected when they do not provide coverage, even if it is necessary to fall back to another matching algorithm with greater coverage for when they do not.

4.2 Relevance

We evaluate the relevance of the predictions using a suite of measures similar to that used by White et al. [2009] for a similar task. We used multiple measures to provide more complete information about where the algorithms were performing well and where they were performing poorly.

Evaluation measures in similar settings such as the KDD Cup 2005 [Li et al. 2005] often use the F1 measure (also known as *test accuracy*), which computes the harmonic mean of precision and recall. However, the work of Phan et al. [2007] demonstrated that long queries typically have more narrow query intent associated with them. The practical use of successful label prediction technology would most likely be in providing a surrogate for query intent and/or in reranking search results. It is rare that more than 10 result snippets (i.e., Web page titles, summaries, and URLs) will be viewed by a user before reissuing a different query. Thus there must be a focus on early precision of the category label prediction so that a search engine could identify pages that match the top one or at most two categories, and promote them higher in the rankings. For any query, the recall depth is computed based on the number of predicted labels. In addition to computing F1, we devised three different precision measures to measure precision at the top of the predicted label ranking—*P@1(top)*,

P@1(any), and *P@3*—and two additional measures that also captured the relevance of highly-ranked items: *mean reciprocal rank* and *normalized discounted cumulative gain*. These metrics were computed over queries for which labels could be assigned.

P@1(top). The first measure P@1(top) required that the top predicted category label pl_1 for an *alqt* matched its top actual label l_1. If so, the prediction algorithm would be given a score 1, and 0 otherwise. The scores over all *alqt* in the test sets were then summed and averaged to provide a final P@1(top) score. This measure is a very high standard, which basically requires a direct match between the top prediction and the top label.

P@1(any). The second measure P@1(any) allowed any predicted category label $pl_1, \ldots, pl_p$ for an *alqt* to be compared with its top actual label l_1. If any of $pl_1, \ldots, pl_p$ matched l_1, the prediction algorithm would be given a score 1, and 0 otherwise. As before, scores were summed and averaged to compute final P@1(any) scores for each of the algorithms and test sets. This measure assumes that the top-ranked label l_1 of an *alqt* is the most important label, and that predicting this at all is valuable.

P@3. The third measure P@3 compared the top predicted category label pl_1 for an *alqt* with any of its top three actual labels l_1, l_2, l_3. If there was a match, the prediction algorithm would be given a score 1, and 0 otherwise. Scores were summed and averaged to compute final P@3 scores as before. This measure assumes that at most one label prediction would be used in a real system, but that helping get any of the three dominant intents correct would be useful.

Mean reciprocal rank. A standard alternative measure used often in Web search evaluation tasks is mean reciprocal rank (MRR); for example, Chowdhury and Soboroff's investigation [2002] reported its use. To compute this measure in our context, the top actual category label l_1 from an *alqt* was compared progressively down the ordered list of predicted category label predictions $pl_1, \ldots, pl_p$. If l_1 matched pl_i, the score assigned was the reciprocal of the prediction rank position—$1/i$, and 0 otherwise. Scores were summed and averaged to compute MRR overall. This measure rewards prediction algorithms that get the most likely category label towards the top of the list of their predictions. In the example for the query [staining a new deck] in Section 3.4, computing the MRR (of 0.5) would involve first comparing the top-ranked predicted label (*Top/Business/Chemicals/Coatings_and_Adhesives*) against the top-ranked ground truth label (*Top/Shopping/Home_and_Garden/Home_Improvement*) before finding a match with the second-ranked ground truth label.

nDCG. The final measure was a variant on nDCG [Järvelin and Kekäläinen 2002]. nDCG biases towards the early retrieval of highly-relevant documents, although it also includes a recall component to the calculation. In our case the documents are category labels, such that the list of actual labels from an *alqt* is considered an ideal vector, with each actual label given a relevance score of 1.

The list of predicted labels $\{pl_1, \ldots, pl_p\}$ is then compared to the ideal vector, and a discounted cumulative gain score computed, using a standard log_2 discount factor. The twist on standard computation of nDCG was in restricting the depth of the comparisons between the two label vectors to the minimum length of the two. The score was normalized by dividing it by the maximum possible value that could be obtained to this depth, and the results summed and averaged as before.

In the next section we present the findings of our experiment.

5. FINDINGS AND DISCUSSION

The coverage calculation and the relevance measures described in Section 4 were used to evaluate the performance of each of the similarity matching functions. The findings are reported in Table III(a). These results were calculated by first sampling 10 random sets without replacement from $trails_{te}$, where each random set consisted of 100,000 *alqt*. Evaluation and coverage measures were computed over each set, and the results averaged. For simplicity, all evaluation measures have been scaled to report in the range (0, 100). The maximum of the standard errors between the means is reported also.

We observe that the term dropping and named entities approaches both suffer from the problem that the historical query set used may not permit any matching queries to be identified. Thus their coverage level is roughly a half and a third respectively of the other methods. Although precision is low, long and rare queries are one of the most challenging query segments for search engines, and being able to accurately label queries for a subset of queries (one-in-five times in the case of BM25 (T=5)) could still result in user benefit. We explore the extent of this benefit in the next section. The low standard error values suggest that the differences between matching algorithms for each of the metrics were statistically significant. This was verified with independent-measures analysis of variance (ANOVA) tests followed by *post-hoc* Tukey tests across all algorithm pairs (all $p < 0.001$). Given the large sample sizes, all differences between algorithms across all measures reported in this section are statistically significant at $p < 0.001$.

Language modeling techniques are known to outperform BM25 in the case of ad-hoc information retrieval [Bennett et al. 2008]. However, the results suggest that LM-based retrieval is not well-suited for the task of retrieving queries, performing worse than all other approaches we tried. From parameter sweeps on training data we identified a value of five as the best setting for the Dirichlet parameter (μ). Further, as we increased μ, that is, we interpolated to a greater extent with the background collection model, we noticed that the LM technique had a tendency to retrieve queries of progressively greater length than the original. The lower performance is therefore not surprising: queries even longer than the original are likely even rarer, and have a lesser chance of being assigned specific (or high-quality) ODP category labels.

The BM25 approaches score significantly better across all measures than the other approaches, with the retrieval threshold of top-three performing marginally better than top-ten. The work of Najork et al. [2007] suggests that

Table III. Coverage and Relevance of Query Label Prediction Using Different Similarity Matching Algorithms

Algorithm	Cov(%)	P@1(top)	P@1(any)	P@3	MRR	nDCG	F1
	(a) Exact query equivalence						
Term dropping	41.2	10.1	12.1	18.4	15.7	11.3	12.0
Named entities	30.5	9.5	11.4	18.4	15.7	10.6	10.9
LM (T=3)	100	6.7	8.2	13.5	10.3	7.2	8.3
LM (T=10)	100	8.1	9.5	16.8	14.4	9.5	11.8
BM25 (T=3)	100	13.2	16.0	24.9	21.0	15.6	15.9
BM25 (T=10)	100	12.7	15.2	24.1	21.4	14.1	15.1
Standard errors		≤ 0.06	≤ 0.04	≤ 0.05	≤ 0.04	≤ 0.07	≤ 0.07
	(b) Normalized query equivalence						
Term dropping	45.1	10.1	12.1	18.4	15.7	11.2	12.4
Named entities	35.6	9.3	11.3	17.5	15.8	10.4	11.8
LM (T=3)	100	6.9	8.3	14.0	10.8	7.9	8.6
LM (T=10)	100	8.6	10.1	17.2	14.7	9.6	10.7
BM25 (T=3)	100	13.7	16.3	25.4	21.4	15.2	16.6
BM25 (T=10)	100	13.1	15.6	24.5	21.8	14.5	15.8
Standard errors		≤ 0.05	≤ 0.03	≤ 0.04	≤ 0.03	≤ 0.04	≤ 0.06
	(c) Normalized query equivalence and ≥ 5 *clicks*						
Term dropping	49.7	15.5	19.4	25.8	22.0	17.4	17.0
Named entities	36.6	13.2	16.7	23.1	21.0	15.0	14.6
LM (T=3)	100	9.6	12.1	17.7	13.9	11.1	10.2
LM (T=10)	100	12.3	15.2	23.1	19.5	14.2	14.0
BM25 (T=3)	100	19.5	24.4	33.3	28.1	22.0	23.1
BM25 (T=5)	100	22.3	26.1	35.6	32.0	24.2	25.8
BM25 (T=10)	100	19.3	23.9	33.5	29.2	21.5	23.2
Standard errors		≤ 0.02	≤ 0.02	≤ 0.02	≤ 0.03	≤ 0.03	≤ 0.04
	(d) Normalized query equivalence, ≥ 5 *clicks, all report matches*						
Term dropping		12.9	14.8	23.6	20.3	14.0	15.0
Named entities		12.4	14.5	22.1	18.6	13.5	14.6
LM (T=3)		8.8	10.6	17.4	13.5	10.0	10.6
LM (T=10)		11.0	13.1	21.6	18.4	12.3	13.3
BM25 (T=3)		17.0	20.3	30.8	26.1	18.8	20.5
BM25 (T=5)		20.8	23.4	33.1	28.4	20.1	25.3
BM25 (T=10)		15.9	19.0	29.7	26.1	17.6	19.5
Standard errors		≤ 0.05	≤ 0.03	≤ 0.04	≤ 0.03	≤ 0.04	≤ 0.05

BM25F (a more sophisticated version of BM25 that can assign different weights to different document fields) is better for ranking more specific queries than a range of other features for ad hoc information retrieval in a Web setting. Although we are using BM25 only (and our "documents" are just queries without multiple fields), our queries are long and likely to be more specific. It may be that the behavior of the ranking characteristics of BM25 observed by Najork and colleagues also applies in our scenario and therefore leads to better performance.

5.1 Normalized versus Exact Query Equivalence

As discussed in Section 3.1, we prepared the trails data both exactly as recorded and using a query normalization algorithm. To examine the impact that query normalization had, we ran the same experiment reported above over a different

sampling of 10 random sets without replacement from $trails_{tn}$, where each random set consisted of 100,000 *alqt*. Our hypothesis was that normalized query data would lead to an improvement for two reasons: at some level there should be more data to use per *alqt* if different queries now become equivalent. Secondly, the removal of erroneous differences appearing due to different word ordering, punctuation, and pluralization are likely to outweigh any increase in query conflation given the length of the queries. The results of this analysis are summarized in Table III(b).

Fairly consistently, scores increased for all similarity matching algorithms across all measures, and the standard errors decreased. Thus from hereon, we used only $trails_{tn}$ as the source of sample *alqt* for our experiments.

5.2 Exploring the Retrieval Threshold

Given the strong performance of the BM25 algorithm, we chose to explore how modifying the BM25 retrieval threshold (i.e., the number of historical queries used to predict category labels) affected its prediction performance. We reran the experiment on one of our randomly-selected normalized sets of 100,000 queries, and varied the retrieval threshold for BM25 from 1 to 20 similar queries. In Figure 2 we show the mean average F1 score for the algorithm at each retrieval threshold. From this analysis, it appears that setting a retrieval threshold of five maximizes the F1 relevance score; all other evaluation measures followed a similar trend. With less than five similar queries the algorithm may lack sufficient labeling information to make an accurate prediction. More than five similar queries there may be noisy labels that reduce the accuracy of the prediction.

5.3 Improved "Judgment" Confidence

One observation we made on examining the predictions where our relevance scores were poor was that due to the sparsity of click data in a subset of the *alqt* test set which constituted our ground truth, the actual labels periodically looked of poor quality. Manual inspection of the query and the predicted labels suggested that our predictions might in fact be good. We hypothesized that because an *alqt* might represent only one occurrence of an unseen query, and have just a single user, using a single search engine, and visiting one or two Web pages, these extremely sparse trails might be distorting the evaluation measures on the quality of predictions.

To investigate this, we reran the evaluation of Section 5.1, holding out queries from the test set $trails_{tn}$ where fewer than five clicks were associated with any *alqt*. The results of the evaluation for all approaches are shown in Table III(c). Note that the coverage values increase for both term dropping and named entities algorithms, since the test set size has decreased. The scores for almost all algorithms increase substantially across all relevance measures. In particular, we see the BM25 (T = 5) scores increasing in the region of 26%. The relative performance ordering of algorithms remained the same as that reported in analysis earlier in the section.

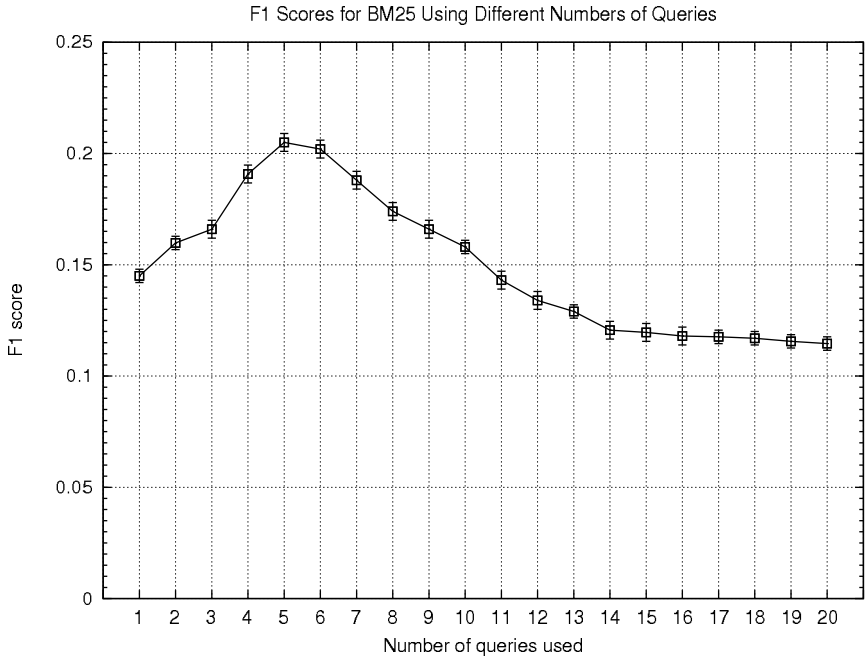


Fig. 2. Exploring the retrieval threshold for BM25 (± SE).

5.4 Overlap Comparison

To provide a fairer examination of the relative relevance performance of the different algorithms, we show in Table III(d) the scores on a subset of queries for which all algorithms were able to predict labels. Although the ten sets of 100,000 sampled unseen queries were obviously of equal size, the subsets from each of these differed marginally. The average size of the overlap subsets was 14,191 *alqt* (a total of 141,912 *alqt* overall). The coverage column is dropped, because all algorithms had 100% coverage. The relative performance ordering once again remained the same.

5.5 Effect Size

All of the observed differences in the means between algorithms were statistically significant with paired measures t-tests (at $p < 0.001$). This was to be expected given the large sample sizes. The final analysis we performed was to ensure that the observed differences between the algorithms were meaningful. We used Cohen's d tests to determine the effect size of each between-algorithm comparison [Cohen 1988]. Table IV shows the obtained d-values. The effect size is small to medium, although it is largest when comparing the BM25 algorithms with comparator algorithms; this signifies that BM25 had the largest treatment effect and its prediction performance is most different from the other matching algorithms under consideration.

The findings presented in this section demonstrate that the BM25 matching algorithm outperforms the other algorithms tested. In the next section we

Table IV. Effect Size for Between-Similarity Matching Algorithm Comparisons

Algorithm	TD	NE	LM(3)	LM(10)	BM25(3)	BM25(5)	BM25(10)
TD	–	.13	.12	.11	.22	.32	.28
NE		–	.11	.10	.23	.31	.30
LM(3)			–	.09	.29	.30	.26
LM(10)				–	.24	.32	.27
BM25(3)					–	.20	.09
BM25(5)						–	.17
BM25(10)							–

evaluate the potential effectiveness of applying the query labels generated by this algorithm for the task of improving retrieval performance.

6. APPLICATION

Long and unseen queries are an important challenge for Web search engines. Such queries constitute around 16% of all unique queries that such engines receive.[3] The findings presented in the previous section demonstrate that it is possible to assign the correct top-ranked label to a long and unseen query around one in five times. While promising, this does not provide insight into the utility of assigning category labels to long and unseen queries.

To address this concern, we evaluated one possible application of query labeling—result reranking. We used the query labels to rerank the top search results generated by a popular and effective ranking algorithm for a set of long and unseen queries. We randomly sampled a set of 75 queries from months of query logs from Microsoft's Bing search engine. These queries contained more than four tokens (i.e., were long per our definition), were normalized, and were unseen (i.e., appeared in $trails_{tn}$ but not in $trails_{hn}$). We ranked Web search results using the popular BM25F algorithm [Robertson et al. 2004], and then reranked the top-n results using two techniques:

ODP. The top-results were reranked using the query labels and ODP category labels assigned to each of those results per the technique proposed by Shen et al. [2005], used to label query trails described earlier in this article. We began by initializing the score for all n search results to zero, effectively giving them all the same initial rank. For each result, we stepped through the labels assigned to the current long and unseen query using the BM25 labeling algorithm, which outperformed the other algorithms in the analysis presented in the previous section. We compared the result's ODP category label against each of the current labels for the query. If there was an exact match between the query label and the result label we incremented the score for that result by one. If there was an inexact match (i.e., category backoff was required), then

[3]Note that an automated analysis of a set of 1,000 randomly-selected long and unseen queries revealed that only 11.8% included spelling errors. The analysis involved submitting each query to the Microsoft Bing search engine and monitoring whether the spelling suggestion was shown on the search engine result page. This signifies that most queries were unique because they expressed search intent in a unique way and were not simply incorrectly-spelled variants of seen queries.

we incremented the result's score by the number of levels in the result's category label when a match was made, divided by the total number of category levels for the result label. This penalized results for only broadly matching the query intent. The top-n results were then sorted based on these scores and the rank order of those results with a score of zero at the bottom of the ranked list were randomly permuted.

Comb. We also combined all pairs of approaches and all three approaches using a rank aggregation technique known as the Borda Count method [Dwork et al. 2001]. Specifically, our approach involved reranking results based on the average of the rank positions of each result, when each method was applied independently. For example, if a result was ranked at position five in the ODP-based ranking and position three in the BM25F-based ranking then the final rank position of that result would be four. All results were ordered based on their average ranks from the two methods.

For each of the 75 queries in our set, we obtained human relevance judgments on a six-point scale—*Bad*, *Poor*, *Fair*, *Good*, *Excellent*, and *Perfect*—for each of the top-50 search results obtained by the BM25F algorithm. We then computed the nDCG scores at positions 1, 3, and 10 for the original ranking and the two reranking strategies described above, and report the average values in Table V averaged across 100 experimental runs (to handle the random permutation). We experimented extensively with reranking the top-10, 20, . . . , 50 results and using the top-1, 2, . . . , 10 query labels. The reranking performance was highly sensitive to the number of results reranked and the number of query labels used. The reported values in Table V are for reranking the top ten results and using the top three predicted labels. These were the best performing parameter settings, perhaps because: (i) BM25F had already placed relevant results in the top ten, reducing the likelihood that erroneous results would be introduced by the reranking, and (ii) selecting three query labels provided a good sense of query intent but not too much information that might introduce noise. A similar pattern was observed while exploring the BM25 threshold in Section 5.2.

To address potential concerns about the need for labeling, we also rerank the BM25F results using only the result page click-through logs for queries most similar to each long and unseen query. This bypassed the labeling process and allowed us to investigate the utility offered by query labeling. To do this, we obtained one month of click-through logs from the Microsoft Bing search engine during December 2009 from the English-speaking United States ISO locale. We ran each query through the BM25 (T = 5) labeling algorithm described in Section 3.3.4 and obtained the top-five most similar queries that appeared in $trails_h$ and in the click-through logs. The BM25 (T=5) algorithm was shown to perform best in finding similar queries for the labeling experiments presented in the previous section and hence seemed appropriate for this task. The clicked URLs and their frequency counts for each of these five queries were retrieved from the logs and their frequency counts aggregated to obtain a click-based final result ranking for each long and unseen query. The top-50 BM25F search results were reordered according to these frequency counts. Results for which

Table V. nDCG for BM25F, Click Reranking, ODP Reranking, and Combinations

Method	nDCG@1	nDCG@3	nDCG@10
BM25F (baseline)	43.7	45.4	46.7
Click	**45.5**	**46.5**	47.2
ODP	**46.8**	**47.9**	**48.9**
BM25F+Click	**46.7**	**47.0**	47.1
BM25F+ODP	45.3	46.2	47.9
Click+ODP	**47.7**	**47.8**	**48.2**
All (BM25F+ODP+Click)	**46.4**	**47.7**	**49.0**

we did not have click-through data were assigned a score of zero and placed at the bottom of the ranked list in a randomly permuted order similar to how we handled ODP reranking. The results for click-through analysis are also shown in Table V.

The findings suggest that reranking based on the ODP labels alone can outperform BM25F and click-through, at rank positions one, three, and ten. Combining features appears to help, at least leading to a slight increase in retrieval performance over the worst-performing algorithm of the combination. Statistically significant nDCG differences over the BM25F baseline—using paired *t*-tests—are marked in Table V in italics (for $p \leq .05$) and bold (for $p \leq .01$). All differences between ODP and Click-through were significant (all $t(74) \geq 2.38$, all $p \leq .02$). The strong performance of the ODP labeling (especially over click-through) was promising as it shows that there may be relevance signal in the labels that could be leveraged to improve the retrieval performance of search engines for long and unseen queries and that the trail-based labeling adds value over the click-through alone. In future work we will investigate the value of reranking results generated by more sophisticated ranking algorithms, such as those that use machine learning techniques and more sophisticated ways to perform the reranking beyond simple rank aggregation.

7. CONCLUSIONS AND FUTURE WORK

Our research makes a number of contributions. We developed an experimental framework to investigate some challenges in labeling long and rare queries, where these queries have never previously been issued to a Web search engine. Such queries are difficult for search engines because of the lack of interaction log data available for them. The framework makes use of query trails so that evaluation of predicted labels can be performed without costly human involvement.

We divided the label prediction problem into separate parts: finding similar queries from past user data, combining the real labels from aggregated query trails containing these queries, and generating a ranked list of labels as a prediction for the query. Our investigation focused on comparative evaluation for the first task—finding similar queries. We held the other aspects of our experiment constant to eliminate sources of difference between approaches.

For the task of label prediction, the order of performance from best to worst was probabilistic ranking (using BM25), term dropping, named entities, and lastly (somewhat surprisingly given its general utility for ad hoc information retrieval) language modeling (using Indri). We investigated retrieval thresholds for the BM25 approach, and found that a five-query threshold was the optimal value. Across a random set of 100,000 long and unseen queries, this approach achieved an F1 score of 20.5, and when we imposed confidence thresholds on the test set truth labels (by requiring five or more clicks in the query trail), it improved to 25.8. Even requiring strict matching between only the top predicted label and the top real label achieved a P@1 score of over 22. In other words, in better than one in five cases we predict the label perfectly. However, this means there is still plenty of scope for future research in identifying other factors in labeling long and rare queries.

We found that normalizing queries improved performance for all algorithms. Normalizing queries may reduce errors arising from minor inconsistencies in past user data. We recommend it in all cases; however, some approaches (such as named entities) must extract data while still in the original term order. Overall, mining query trails from past user data provides a useful source for predicting the category labels for never-before-seen long and rare queries. The techniques evaluated in this article were all simple and scalable, and could be implemented as an additional service in a search engine that would map an incoming long and unseen query to category labels and use the labels to enhance the search experience.

We also showed that the query labels have utility for at least one application: reranking search results. We used the query labels as a richer representation of query intent and applied them to the task of reordering the top retrieved results served by a popular search algorithm (BM25F). We demonstrated that the application of the labels for result reranking in combination with BM25F led to a significant increase in retrieval effectiveness over BM25F alone. Search engines may wish to employ richer representations of long and unseen queries to facilitate result reranking, tailored ranking algorithms, and other special treatment options, such as richer result snippets.

Future work includes greater success and failure analysis. For example, are there query patterns for which particular algorithms consistently do well or poorly on, or are there patterns in the trails (e.g., cycles of repeat visits to the same URLs) that lead to improved or degraded performance? The overlap comparison indicates that the named entities approach noticeably differs from the term dropping approach (they produce matches in common for less than 50% of queries). This observation supports our belief that distinct query patterns exist with long queries, and may be susceptible to differential treatment if the patterns are consistent. The effect of removing stop words from queries both in the past usage data and in test queries would be interesting to explore for the BM25 approach. Another important area is the effect of applying different approaches in combining the real labels and in generating the ranked list of labels, rather than simple count aggregation methods. Determining the limitations of log mining for label selection would also be valuable.

ACKNOWLEDGMENTS

We would like to thank our colleagues Nick Craswell, Mukund Narasimhan, Milad Shokouhi, Paul Viola, and Haizheng Zhang for the discussions about the challenges of analyzing long queries. We would particularly like to thank our colleagues Silviu-Petru Cucerzan for providing us with access to his named entity identification technology, Gheorghe Muresan for access to his query normalization code, Krysta Svore for assistance with ranking algorithms, and Stefan Savev for help with F#.

REFERENCES

AGICHTEIN, E., BRILL, E., AND DUMAIS, S. 2006. Improving Web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 19–26.

ALLAN, J., CALLAN, J., CROFT, W. B., BALLESTEROS, L., BROGLIO, J., XU, J., AND SHU, H. 1997. Inquery at TREC-5. In *Proceedings of the 5th Text Retrieval Conference (TREC)*. NIST, 119–132.

ALLAN, J. AND RAGHAVAN, H. 2002. Using part-of-speech patterns to reduce query ambiguity. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 307–314.

BEITZEL, S. M., JENSEN, E. C., FRIEDER, O., LEWIS, D. D., CHOWDHURY, A., AND KOLCZ, A. 2005. Improving automatic query classification via semi-supervised learning. In *Proceedings of the International Conference on Data Mining*. 42–49.

BEITZEL, S. M., JENSEN, E. C., LEWIS, D. D., CHOWDHURY, A., AND FRIEDER, O. 2007. Automatic classification of Web queries using very large unlabeled query logs. *ACM Trans. Inform. Syst. 25,* 2.

BENDERSKY, M. AND CROFT, W. B. 2008. Discovering key concepts in verbose queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, 491–498.

BENNETT, G., SCHOLER, F., AND UITDENBOGERD, A. 2008. A comparative study of probabalistic and language models for information retrieval. In *Proceedings of the 19th Annual Australasian Database Conference*. 65–74.

BENNETT, P. N., SVORE, K., AND DUMAIS, S. 2010. Classification-enhanced ranking. In *Proceedings of the 19th International Conference on World Wide Web (WWW'10)*.

BILENKO, M. AND WHITE, R. W. 2008. Mining the search trails of surfing crowds: Identifying relevant Web sites from user activity. In *Proceedings of the 17th Annual Conference on the World Wide Web*. 51–60.

BOLLEGALA, D., MATSUO, Y., AND ISHIZUKA, M. 2007. Measuring semantic similarity between words using Web search engines. In *Proceedings of the 16th International Conference on the World Wide Web*. ACM, New York, 757–766.

BRODER, A. Z., FONTOURA, M., GABRILOVICH, E., JOSHI, A., JOSIFOVSKI, V., AND ZHANG, T. 2007. Robust classification of rare queries using Web knowledge. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 231–238.

CALLAN, J. P., CROFT, W. B., AND BROGLIO, J. 1995. TREC and tipster experiments with inquery. *Inform. Process. Manage. 31*, 3, 327–343.

CHIRITA, P. A., NEJDL, W., PAIU, R., AND KOHLSCHÜTTER, C. 2005. Using ODP metadata to personalize search. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'05)*. 178–185.

CHOWDHURY, A. AND SOBOROFF, I. 2002. Automatic evaluation of world wide Web search services. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 421–422.

COHEN, J. 1988. *Statistical Power Analysis for the Behavioral Sciences*, 2nd Ed. Lawrence Earlbaum.

CUCERZAN, S. 2007. Large-scale named entity disambiguation based on Wikipedia data. In *Proceedings of EMNLP-CoNLL*. 708–716.

DWORK, C., KUMAR, R., NAOR, M., AND SIVAKUMAR, D. 2001. Rank aggregation methods for the Web. In *Proceedings of the 10th International Conference on World Wide Web (WWW'01)*. 613–622.

GRAVANO, L., HATZIVASSILOGLOU, V., AND LICHTENSTEIN, R. 2003. Categorizing Web queries according to geographical locality. In *Proceedings of the 12th ACM CIKM Conference on Information and Knowledge Management*. 325–333.

JÄRVELIN, K. AND KEKÄLÄINEN, J. 2002. Cumulated gain-based evaluation of IR techniques. *ACM Trans. Inform. Syst. 20*, 4, 422–446.

KARDKOVÁCS, Z. T., TIKK, D., AND BÁNSÁGHI, Z. 2005. The Ferrety algorithm for the KDD Cup 2005 problem. *SIGKDD Explor. 7*, 2, 111–116.

KUMARAN, G. AND ALLAN, J. 2007. A case for shorter queries, and helping users create them. In *Proceedings of the HLT-NAACL*. 220–227.

KUMARAN, G. AND ALLAN, J. 2008. Effective and efficient user interaction for long queries. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 11–18.

KUMARAN, G. AND CARVALHO, V. R. 2009. Reducing long queries using query quality predictors. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York, In press.

LEASE, M., ALLAN, J., AND CROFT, W. B. 2009. Regression rank: Learning to meet the opportunity of descriptive queries. In *Proceedings of the 31st European Conference on Information Retrieval*. Springer-Verlag, 90–101.

LI, X., WANG, Y.-Y., AND ACERO, A. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 339–346.

LI, Y., ZHENG, Z., AND DAI, H. K. 2005. KDD CUP-2005 report: Facing a great challenge. *SIGKDD Explor. 7,* 2, 91–99.

METZLER, D., DUMAIS, S., AND MEEK, C. 2007. Similarity measures for short segments of text. In *Proceedings of the 29th European Conference on Information Retrieval*. 16–27.

NAJORK, M. A., ZARAGOZA, H., AND TAYLOR, M. J. 2007. HITS on the Web: How does it compare? In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 471–478.

PHAN, N., BAILEY, P., AND WILKINSON, R. 2007. Understanding the relationship of information need specificity to search query length. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 709–710.

PONTE, J. M. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 275–281.

QIU, F. AND CHO, J. 2006. Automatic identification of user interest for personalized search. In *Proceedings of the 15th International Conference on World Wide Web (WWW'06)*. 727–736.

ROBERTSON, S., WALKER, S., JONES, S., HANCOCK-BEAULIEU, M., AND GATFORD, M. 1994. Okapi at TREC-3. In *Proceedings of the 3rd Text REtrieval Conference (TREC'94)*.

ROBERTSON, S., ZARAGOZA, H., AND TAYLOR, M. 2004. Simple BM25 extension to multiple weighted fields. In *Proceedings of the 13th ACM CIKM Conference on Information and Knowledge Management*. ACM, New York, 42–49.

SHEN, D., PAN, R., SUN, J.-T., PAN, J. J., WU, K., YIN, J., AND YANG, Q. 2005. Q^2C@UST: Our winning solution to query classification in KDDCUP 2005. *SIGKDD Explor. 7*, 2, 100–110.

SHEN, X., DUMAIS, S., AND HORVITZ, E. 2005. Analysis of topic dynamics in Web search. In *Proceedings of the 14th International Conference on the World Wide Web*. 1102–1103.

STROHMAN, T., METZLER, D., TURTLE, H., AND CROFT, W. B. 2005. Indri: A language-model based search engine for complex queries (extended version). IR 407, U. Massachusetts.

VOGEL, D. S., BICKEL, S., HAIDER, P., SCHIMPFKY, R., SIEMEN, P., BRIDGES, S., AND SCHEFFER, T. 2005. Classifying search engine queries using the Web as background knowledge. *SIGKDD Explor.* 7, 2, 117–122.

VOORHEES, E. M. 1994. Query expansion using lexical-semantic relations. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 61–69.

WHITE, R. W., BAILEY, P., AND CHEN, L. 2009. Predicting user interests from contextual information. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. In press.

WHITE, R. W. AND DRUCKER, S. 2007. Investigating behavioral variability in Web search. In *Proceedings of the 16th International Conference on the World Wide Web*. ACM, New York, 21–30.

ZHAI, C. AND LAFFERTY, J. 2001. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 334–342.

Received June 2009; revised February 2010, April 2010; accepted May 2010