Reranking Search Results for Sparse Queries

Elif Aktolga and James Allan
Center for Intelligent Information Retrieval
Department of Computer Science
University of Massachusetts Amherst
Amherst, Massachusetts
{elif, allan}@cs.umass.edu

ABSTRACT

It is well known that clickthrough data can be used to improve the effectiveness of search results: broadly speaking, a query's past clicks are a predictor of future clicks on documents. However, when a new or unusual query appears, or when a system is not as widely used as a mainstream web search system, there may be little to no click data available to improve the results. Existing methods to boost query performance for sparse queries extend the query-document click relationship to more documents or queries, but require substantial clickthrough data from other queries.

In this work we describe a way to boost rarely-clicked queries in a system where limited clickthrough data is available for all queries. We describe a probabilistic approach for carrying out that estimation and use it to rerank retrieved documents. We utilize information from co-click queries, subset queries, and synonym queries to estimate the clickthrough for a sparse query. Our experiments on a query log from a medical informatics company demonstrate that when overall clickthrough data is sparse, reranking search results using clickthrough information from related queries significantly outperforms reranking that employs clickthrough information from the query alone.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Search process

General Terms: Experimentation, Algorithms

Keywords: Sparse Queries, Reranking, Query Log Mining, Sparse Clickthrough Data, Query Selection

1. INTRODUCTION

Clickthrough data from query logs is widely used to improve document ranking [1, 8, 9, 10, 14, 15, 25]. But how does it work for new or unusual queries in a search system? Such (sparse) queries are problematic because little or no clickthrough information is available for them in query logs. This situation can be attributed to three causes: (1) either the query is unpopular or new and therefore very few searches have been issued with this query; (2) the query is not unpopular but the displayed ranking of the search results does not fulfill users' information needs to the extent that results are rarely clicked; or (3) the query is not unpopular but the retrieved list of documents is of poor quality so that reranking with the same set would not improve the results. Mostly, such queries can be boosted with their own sparse clickthrough data if available [23], or information from other more popular queries such as co-click queries may be used to enrich the results [8, 25]. Existing approaches however require substantial clickthrough data from *somewhere* for solid improvements due to noise that is introduced in gathering the additional data [10, 23]. In this context, our work aims at answering the following question: how can we rerank search results for rare or sparse queries in a system for which *limited clickthrough data* is available?

Our aim is in particular to achieve a better ranking of documents for underrepresented or sparse queries of types (1) and (2) described above. We do not deal with case (3) since this situation must be approached with a better retrieval algorithm and not with reranking.

We deal with issues (1) and (2) indirectly by estimating how likely a document is to be clicked given a query by using a language modeling framework [19]. Our hypothesis is that using clickthrough data from *related* queries for sparse queries aids in achieving a better ranking than solely using the query's own sparse clickthrough data. The difficulty of the task arises from the fact that little or no clickthrough data is available not only for the query in question, but also for *all other queries* in the query log. In this situation, we cannot afford to use an approach that introduces too much noise.

For our task, we employ queries related to the query whose results are to be reranked. They come from one of the following three sources: 1. *Similar queries:* these share at least one co-click with the original query, which is a well-known feature from previous work [8, 10, 13, 23]; 2. *Subset queries:* these are contained in the original query as an n-gram sequence; 3. *Synonym queries:* these are lexically, semantically, or syntactically related to the original query. We use the clickthrough data of these related queries in our models.

We test our models on queries from a medical query log that was obtained from a medical informatics company, UpToDate. We also use their Lucene-based search system that yields the initially ranked search results for a query. The users of this system are doctors and nurses in practices and hospitals – i.e., specialists in their domain. Therefore, the medical queries are very specific to the domain. In such a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.
Copyright 2011 ACM 978-1-4503-0717-8/11/10 ...$10.00.

system, sparsity of clickthrough data is experienced more strongly than in general search engines that quickly acquire vast amounts of queries per day.

Our results show that all reranking models outperform the baseline for which clickthrough-boosting is employed with information from the original query only. Further, we observe some interesting trends among the reranking models when reducing clicks in the dataset. Reranking with subset queries and a merged version that uses all related queries perform best, which is expected.

Our results are specific to a particular medium-scale search system, though we believe they are general enough to apply to any similar setting. We hypothesize that the same ideas will extend to general search engines with much greater levels of activity, but do not have access to the resources needed to test that belief.

This paper is organized as follows: the next Section 2 reviews relevant related work in the area. In Section 3 we detail the empirical setting that motivated us for this research and that constrains our possible solutions. Then, we describe the baselines and reranking models that use clickthrough data from related queries in Section 4. This is followed by the experiments and the results in Section 5. Finally, we conclude our work in Section 6.

2. RELATED WORK

It is well known that by means of clickthrough data search results can be reranked to significantly improve the results [1, 8, 9, 10, 14, 15, 25]. However, clickthrough data sparsity poses a problem [6, 8, 10, 13], since many queries often have a small number of clicks, which makes them incomplete. Indeed, queries often have no clicks at all. Gao et al. [10] apply two smoothing methods to overcome the data sparsity problem, one of which is based on Good Turing smoothing. This is achieved by smoothing the features of 0-click queries with features of 1-click queries. The other 'smoothing' method is based on query clustering on the clickthrough graph [8]. Here, the query-document click graph is extended with more queries and documents by following links between them. However, in their experiments on a dataset with less clickthrough data available, Gao et al. [10] show that the query clustering approach on the clickthrough graph does not help. This method only works well when substantial clickthrough data is available, since the introduction of additional documents in the click graph is likely to be noisy for queries originally not having a click with such documents.

One similarity between our work and Craswell and Szummer's random walk on the click graph [8] is that we use information from co-click queries to estimate parts of the reranking models – this is equivalent to following a link between a document and a co-click query. However, one major difference is that we do not include new documents in the document list to be reranked, since this is prone to be noisy when clickthrough data is sparse. This means that we do not go beyond the scope of 1-hop co-click queries. Another major difference is that in Craswell et al.'s work the query content is not considered in the reranking process, whereas we also attempt to use information from other related queries for which the subqueries and synonyms of a query have to be analyzed.

There has been a lot of previous research that uses information from co-click queries for reranking [8, 10, 13, 23]. These queries are what we refer to as 'similar queries', and they are one possible source of related queries in our reranking models. Zhao et al. [25] use clickthrough frequencies of subqueries in their reranking approach while calculating the likelihood of a document being retrieved, given a query. Synonym queries on the other hand have been explored in the literature in the form of query reformulation, query expansion and query rewriting [11, 17, 22].

There are several learning to rank approaches that incorporate clickthrough data [1, 9, 20]. Although we employ a simple regression approach to estimate the similarity of some queries (Section 4.3.2), we do not otherwise use learning approaches. For reasons explained in the following section, our empirical evaluation is constrained to use a black-box retrieval system, for which we are combining very few features. A straightforward parameter sweep is as powerful as any learning method in this case.

To the best of our knowledge, although previous approaches have tested their retrieval and reranking algorithms on sparse queries, they have not employed them on sparse queries in a sparse system where the overall clickthrough data available is limited *for all queries*. In our work, we aim at addressing this aspect in particular. We note that such a setting seems unusual, but is common in the early days of any new search system. Indeed, major search engines have the same problem – i.e., they have queries which have rarely been seen before. Further, their total count of such queries is almost certainly vastly greater. However, since these queries are a more noticeable part of the search experience for smaller search engines like that of UpToDate, this work is more valuable in such a setting. Our results will show that 'sparse' means up until queries have close to 500 clicks.

3. EMPIRICAL SETTING

The motivation for this research arose during discussions with the medical informatics company UpToDate. UpToDate hosts medical information that is searched on a daily basis by a large number of physicians. Although it receives a substantial number of queries each day, the volume is minuscule compared to major search engines: our collection contains around 10 million queries per month compared to an estimated more than 10 billion queries handled by Google this past March [7].

The search engine used by UpToDate is based on version 1.4 of the open source Apache Lucene system[1]. It has been extensively tuned over several years, using parameter sweeps, evaluations on subsets of users, careful editing of the hosted information, and human intuition. A user's query is converted into a complex weighted combination of the original query words, synonyms from a controlled vocabulary, and field references. In this work, we incorporate clickthrough information in the Lucene-based ranking, improving retrieval effectiveness to the point that it is exceptionally good. As we show in Section 5.2, the system is extremely accurate, achieving NDCG closing in on 90% on average.

Not surprisingly, there are a number of queries that perform much worse, the 'sparse queries' in particular. The empirical parts of this study are carried out in this environment. Whereas the baseline search system is a carefully crafted Lucene implementation that incorporates clickthrough information for all queries with enough click history, our ex-

[1] http://lucene.apache.org/java/docs/index.html

periments investigate methods for applying clickthrough information for sparse queries in addition.

The goal of our empirical work is to explore the impact of several approaches for using sparse information to improve the ranking of the UpToDate-based system. That means we are constrained to combining a carefully crafted Lucene score with other information – here, clickthrough information from the query and its related queries. We leave for future work approaches that expose the inner workings of the Lucene black-box system and integrate clickthrough information more elaborately.

4. RERANKING METHODS

In this section we first detail two baseline methods before describing the reranking models that incorporate clickthrough data from related queries.

4.1 Baseline System (Luc)

As mentioned before, the query log and search system we have for this work are based on the Apache Lucene system used by UpToDate. Lucene by default employs a combination of the vector space model and the boolean model for retrieval. The system has further been extended and tuned for this work as described in Section 3. We convert this retrieval system's raw output score, $\text{score}_{\text{Luc}}(Q, D)$, to probabilities for our reranking models (Section 4.3) as follows:

$$P_{\text{Luc}}(D|Q) = \frac{\text{score}_{\text{Luc}}(Q, D)}{\sum_i \text{score}_{\text{Luc}}(Q, D_i)} \quad (1)$$

where the denominator of this equation sums over all documents D_i retrieved for the query Q.

4.2 Clickthrough-Boosted Baseline (BoosLuc)

The Clickthrough-Boosted Baseline (BoosLuc) is an extension of the Lucene system in that every query is boosted with its own clickthrough information from the training data. A Lucene retrieved document D for query Q is reranked as follows:

$$P_{\text{BoosLuc}}(D|Q) = \gamma \cdot \frac{c_{\text{CT}}(Q, D)}{c_{\text{CT}}(Q)} + (1 - \gamma) \cdot P_{Luc}(D|Q) \quad (2)$$

where $c_{\text{CT}}(Q, D)$ is the clickthrough count for query Q with document D, $c_{\text{CT}}(Q)$ is the total clickthrough count for Q used for normalization, and $P_{Luc}(D|Q)$ is from Equation 1. γ is defined as follows:

$$\gamma = \frac{c_{\text{CT}}(Q)}{c_{\text{CT}}(Q) + \rho} \quad (3)$$

which is Dirichlet-like smoothing, ensuring heavier weight on the clickthrough part in Equation 2 when more clickthrough data is available for Q, and less otherwise. With less clickthrough data, $P_{Luc}(D|Q)$, which estimates how likely D is to be relevant by content, has a larger impact. We determine ρ with a parameter sweep on the training data.

4.3 Reranking Models

In this section we describe the reranking models for sparse queries that use clickthrough data from related queries. All reranking models have the same base model, but they differ in their source of related queries from which the information is drawn. The base model estimates the relevance of a document D given a query Q as follows:

$$P_{\text{RelQ}}(D|Q) = \alpha \cdot P_{\text{CT}}(D|Q) + (\alpha - 1) \cdot P_{\text{Luc}}(D|Q) \quad (4)$$

where α is Jelinek-Mercer smoothing tuned on training data. The first component $P_{\text{CT}}(D|Q)$ indicates how likely D is to be clicked based on the clickthrough information obtained from related queries:

$$P_{\text{CT}}(D|Q) = \beta \cdot \sum_{Q' \in \mathcal{Q}} P(D|Q') \cdot P(Q'|Q) + (1 - \beta) \cdot \frac{c_{\text{CT}}(Q, D)}{c_{\text{CT}}(Q)} \quad (5)$$

where $\mathcal{Q}$ is a set of related queries that varies depending on the particular model used. β is again determined by means of Dirichlet-like smoothing as in Equation 3, with the parameter ρ replaced with κ. $P(D|Q')$ indicates the likelihood of D to be retrieved by the related query Q':

$$P(D|Q') = \frac{c_{\text{CT}}(Q', D)}{c_{\text{CT}}(Q')} \quad (6)$$

which is a normalized probability based on Q's clickthrough data so that $\sum_D P(D|Q') = 1$.

$P(Q'|Q)$ in Equation 5 stands for the quality of the relation between Q' and Q. Intuitively, Q and Q' are most related if Q' can predict the way Q's Lucene ranked documents are going to be clicked. In other words, given the Lucene ranked list Q_{Luc} of Q and the clickthrough ranked list Q'_{CT} of Q', the two queries are most related if Q_{Luc} ranks documents in the order they are clicked in Q'_{CT}. The list Q'_{CT} is sorted in decreasing order of frequencies $c_{\text{CT}}(Q', D)$ for Q'. This means that ideally exactly those documents in Q_{Luc} should be ranked lowest that were not clicked for Q' and are thus missing in Q'_{CT}. Hence, we compare Q_{Luc} and Q'_{CT} as follows:

$$P(Q'|Q) = NDCG(Q_{\text{Luc}}) \quad (7)$$

where the relevance judgments for NDCG come from the ranking of Q'_{CT}. That is, we map the clicks given by Q'_{CT} to relevance grades, with which Q_{Luc} is evaluated (see Section 5.1.) In particular, here we use NDCG@n, where $n \leq 10$ – the NDCG value at the highest rank available, not exceeding 10. The NDCG values are normalized to probabilities so that $\sum_{Q' \in \mathcal{Q}} P(Q'|Q) = 1$.

Note that with this approach, documents in Q_{Luc} that are ranked lowest and which were not clicked for Q' (absent in Q'_{CT}) will get a relevance grade of 0 in Q_{Luc}. If those documents are perfectly ranked at the bottom of the list, this will not result in a lower NDCG score.

To summarize, Equation 5 has the following consequences in extreme cases:

1. if there is no clickthrough data available *at all* for Q or any $Q' \in \mathcal{Q}$, then the reranking model will not alter the original ranking as given by the Lucene baseline;
2. if there is no clickthrough data available for *any* of the related queries, or no related queries could be found, but there is information for Q, then the reranking model will utilize the clickthrough likelihood of Q and rank exactly like BoosLuc (Equation 2).

In the usual case where some sparse clickthrough data is available for both Q and related queries, the model is a combination of all these components. Each related query contributes to the likelihood of D being relevant for Q. If there is no evidence for $c_{\mathrm{CT}}(Q', D)$ given a particular Q', then this results in D not being boosted by Q'. Neither do we apply a special form of smoothing here, nor do we use clicks with related queries on other documents to boost D, which would introduce noise into the estimation. The idea is that by means of different ways of locating related queries, we can find enough evidence to boost the right documents.

4.3.1 Similar Query Reranking (Sim)

In this model a related query is a similar query Q' that shares at least one co-click on a document with the original query Q. The domain of similar queries therefore consists of all the training queries in the entire 'clickthrough corpus' that have at least one click in common with Q's ranked list. We constrain the clickthrough corpus here to have a limited and meaningful set of similar queries that excludes junk queries. We locate similar queries efficiently via the reverted indexing approach [18]:

Build Reverted Index.

This is a preparation step for which we obtain the clicked document ids for every training query Q from the clickthrough data from which the index is built. The entries look as follows:

$$\mathit{docid}(D) = \{\text{queries that retrieve } D \text{ having } c_{\mathrm{CT}}(Q, D)\}$$

This way we accumulate the queries that were clicked with every *docid* for all the training queries. The clickthrough counts are normalized so that $\sum_i c_{\mathrm{CT_{norm}}}(Q, D_i) = 1$, where each D_i has at least one click with Q.

Retrieve Similar Queries.

This is how similar query lookup is achieved. Given an initial query Q, we retrieve similar queries $Q' \in \mathcal{Q}$ by querying the reverted index with each of the clicked *docids* for Q. This yields a combined list of similar queries with their normalized clickthrough counts. We conflate multiple entries of a similar query Q' in this list to obtain the set $\mathcal{Q}$ by averaging the normalized clickthrough count for each Q'. This is then used to estimate $P(D|Q')$ (Equation 6).

Rank Similar Queries.

The similar queries are ranked in decreasing order of $P(Q'|Q)$ as described in Section 4.3 (Equation 7). By construction, every similar query has a non-zero $P(Q'|Q)$.

4.3.2 Subset Query Reranking (Sub)

Under this model a related query Q' is defined as a subquery of Q. The domain of related queries is therefore the power set of all non-empty n-gram subqueries of Q. Considering subqueries is useful when the original query is too specific, such as in *pediatric migraine headache*, since subqueries such as *migraine headache* or *headache* often capture more general content. We analyzed in our training data how frequently subqueries of sparse queries are clicked. For this, we took all training queries with fewer than 100 clicks, of which there are about 1 million queries, and obtained their subqueries. Table 1 contrasts the number of these training subqueries having clicks in a certain range versus the total number of training queries having click counts in the same range. The numbers show that subqueries of sparse queries are not as sparse as their corresponding super queries (bold entries in 2^{nd} column), with 10,368 of them having at least 100 clicks. But the subqueries are not more popular either, since the numbers in the higher click count bins are comparable (bold entries in 2^{nd} versus 3^{rd} columns).

Table 1: Number of training subqueries having clicks in a certain range whose super queries have less than 100 clicks, versus the number of all training queries having clicks in that range.

clicks	*#Subqueries of sparse queries*	*#All queries*
1-9	108,807	960,423
10-49	28,421	42,915
50-99	6435	7234
100-499	**7291**	**7489**
500-999	**1426**	**1430**
1000-2499	**1077**	**1078**
2500-4999	**369**	**370**
>5000	**205**	**208**

In the subset query ranking model (Sub), we estimate $P(D|Q')$ as described in Equation 6, but for $P(Q'|Q)$, we train a linear regression function with several features on a portion of the training data, since this form of estimation yields better performance with subset queries. As the truth label during training we use $NDCG(Q_{\mathrm{CT}})$ (with relevance judgments from Q'_{CT}) from a different training split. The features are the following:

$\mathit{overlap}(Q, Q')$ number of overlapping terms between the query and subquery

$\mathit{kcc}(Q), \mathit{kcc}(Q')$ Key Concept Classifier (KCC) score [4]

$\mathit{sim}(Q, Q')$ weighted overlapping terms, where weight $w(t)$ for a term t is either the Google n-gram count [5] or KCC(t):

$$\mathit{sim}_{\mathrm{google/kcc}}(Q, Q') = \frac{\sum_{t \in Q'} w(t)}{\sum_{t \in Q} w(t)} \qquad (8)$$

$\mathrm{score_{Luc}}(Q), \mathrm{score_{Luc}}(Q')$ total Lucene retrieval score for query and subquery

$\mathit{popweight}(Q, Q')$ popularity weight of Q' with Q:

$$\mathrm{popweight}(Q, Q') = \frac{c_{\mathrm{CT}}(Q)}{\sum_{Q^* \supset Q'} c_{\mathrm{CT}}(Q^*)} \qquad (9)$$

where $Q^* \supset Q'$ represents queries Q^* that contain Q'. The popularity weight thus contrasts how popular Q' is as a subquery of Q versus being a subquery of other queries, estimated by means of clickthrough counts.

$\mathit{popscore}(Q, Q')$ popularity score of Q' with Q:

$$\mathrm{popscore}(Q, Q') = \frac{\mathrm{popweight}(Q, Q')}{\sum_{Q'' \in 2^Q} \mathrm{popweight}(Q'', Q)} \qquad (10)$$

where Q'' iterates over all subqueries in Q. The popularity score indicates how popular the subquery Q' is as opposed to other subqueries Q'' in Q.

We ensure all features are between 0 and 1 through normalization. For the subset queries model, $P(Q'|Q)$ is then a linear combination of these features with the trained weights. Note that most of these features just described are applicable to subset queries only. We tried training a function for the other reranking models, too, by including $NDCG(Q_{\text{Luc}})$ as an additional feature. This feature proved to be the strongest one for them, which is what we use to estimate $P(Q'|Q)$ for those approaches.

4.3.3 Synonym Query Reranking (Syn)

In Synonym Query Reranking (Syn), a synonym query is syntactically, semantically, or lexically related to the original query. So variations of a query such as singular/plural or paraphrases such as in *hypernatremia* and *high plasma sodium level* are valid synonym queries. As a synonym resource we use a carefully constructed medical vocabulary by deputy editors from UpToDate. It has 24,415 canonical keyword entries with 150,383 synonym entries in total.

Given a query, we construct the set of synonym queries $\mathcal{Q}$ as follows: first, if a canonical keyword entry exists for the query, we use the synonym entries associated with it. If there is no canonical entry for a query (which often happens with very general medical terms), we take the n-gram subset queries of the query and again look for canonical keyword entries for those subset queries. If still no synonym queries could be found after this step, we try to match the original query and its subqueries in the synonym entries, to then include the associated canonical entry and the remaining synonym entries for that query. The found set of synonyms is filtered to exclude 0-click queries. We use at most 10 synonyms from this set.

$P(Q'|Q)$ and $P(D|Q')$ of these synonym queries are then estimated by means of the training clickthrough data as described in Equations 6 and 7, Section 4.3.

4.3.4 Merged Reranking (Merged)

For this model, we merge the different kinds of related queries to one set, i.e.,

$$\mathcal{Q} = \text{similar queries} \cup \text{subqueries} \cup \text{synonyms}$$

For each type of related query we estimate $P(Q'|Q)$ and $P(D|Q')$ the way it was described for the query in Section 4.3 and its subsections. We do not reduce the number of related queries in this process. Therefore, the effect on Equation 5 is that the sum is performed over a larger set of related queries.

The reason for having this merged reranking model is to compensate for a bad choice of related queries for any one of the methods – such as poor synonyms, no similar queries (for 0-click queries), or no subset queries (for original unigram queries).

5. EXPERIMENTS

5.1 Data and Evaluation

Our query log is from UpToDate. The query log has 4 months of data. Tables 2 and 3 contain some statistics about the entire data set. We split the query log into three folds: two training folds for parameter tuning and one test fold on which the results are reported in Section 5.2. The folds are roughly equal in size.

For the experiments, we use 1407 queries together with the training portion of the query log and 972 queries with the test set. To avoid biasing parameters toward certain queries, the training queries used in parameter tuning are *distinct* from the test queries for which the results in this paper are reported. All training and test queries are popular (i.e., each query has at least 500 clicks). We perform the experiments both on these popular queries and we also simulate them as sparse queries by reducing the amount of training click data available to the models. This is done to understand the performance of sparse queries in a *sparse environment.*

Table 2: Some statistics about the query log.

Feature	Quantity
Total # queries	39,248,767
Unique queries	4,896,827
Queries/day	256,528
Unique queries/day	32,005
Queries/month	7,849,753
Unique queries/month	979,365

Table 3: Distribution of clicks for all queries in the training and test data sets.

Click Range	Train	Test
1-9	960423	634761
10-49	42915	28569
50-99	7234	4838
100-499	7489	5221
500-999	1430	956
1000-2499	1078	660
2500-4999	370	170
> 5000	208	74

Each query comes with clicks in the training folds as well as clicks in the test fold. For some of our experiments we use all of the clicks available. In others we reduce the number of clicks available to the algorithms in order to simulate a sparse situation. The reduction of clicks for sparsity is achieved as follows: to have K clicks for a query, we evenly reduce the click count for each document so that the sum of all the clicks for the query roughly equals K (with some small error, due to multiplication with a small fraction). If the total click count for a query is less than K, the clicks are not altered. Therefore, this reduction of clicks preserves the distribution of the click counts. We choose this form of click reduction to preserve the trend in the data, i.e., the most frequently clicked document will still be the same after reduction. Note that clicks can be reduced within a single fold, but can also be reduced across two or three.

Our aim in reranking the Lucene results is to better approximate the ranking as given by the click frequencies from users, similar to Zhao et al's work [25]. Hence we use the clicks in the test fold as a representation of "truth." For the experiments reported in this paper, clickthrough truth comes from the test fold and model estimation is done with the training folds. Note that while the training clicks are reduced in some of the experiments, the clickthrough truth data from the test split is never reduced.

Given a query Q, in order to evaluate its ranked list of n documents R, we use two different measures. One of them uses relevance judgments from clickthrough truth, whereas the other compares the rank correlation to the clickthrough truth list. For the former measure, we use the truth clicks for Q from the test fold as relevance judgments. We do this by mapping the clicks to relevance grades for the well-known NDCG measure (Normalized Discounted Cumulative Gain) [12]:

$$\text{NDCG(Q, R)} = \frac{1}{Z_Q} \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{\log_2(1+i)} \qquad (11)$$

where i is the i^{th} document in R and rel_i is the relevance grade of document i. This measure is normalized with Z_Q so that NDCG=1.0 when ranking is perfect. We would like to be able to uniformly map the clicks to a reasonable range of relevance grades to be used with NDCG for *any given query.* We experimented with various schemes of mapping the clicks for all the queries in the training query log portion to find the most suitable approach. We discovered that we can achieve a reasonable range of real-valued relevance grades roughly between 1 and 4 if we use $\log_{10}(clicks)$ for the mapping.

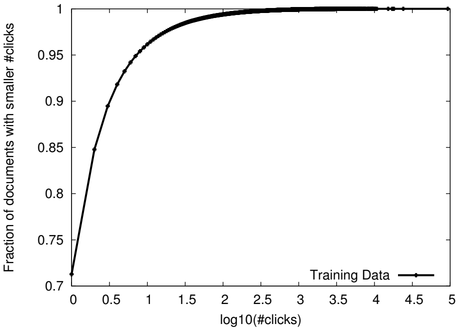

Figure 1: Mapping clickthrough data to NDCG relevance grades for all queries in the training query log.

Table 4: Relevance judgments from the entire test set rounded to discrete values.

Rel. Grade	Test total
1	7121
2	2772
3	1115
4	30

Figure 1 shows the effect of choosing $\log_{10}(clicks)$ with a cumulative distribution function of the clicks on the documents of all the queries in the training data. Note that by this construction relevance grades are real-valued rather than discrete.

Table 4 shows the corresponding relevance judgments for the entire test set. For this, we rounded the real-valued relevance grades to discrete values. The numbers indicate that across the dataset, there are only 30 'perfect' documents, 1115 'excellent' documents, 2772 'fair' documents, and 7121 'bad' documents, which should not be ranked highly.

Therefore, given a query Q and a ranked list R obtained through a reranking method as detailed in Section 4, we evaluate the ranking of each $D \in R$ by means of the *truth clicks* of (Q, D) from the *test fold* mapped to NDCG relevance grades as follows: $\log_{10}(c_{\text{CT}}(Q, D))$. Similarly, in Equation 7 where we estimate the quality of the relation $P(Q'|Q)$ by means of NDCG, the frequencies $c_{\text{CT}}(Q', D)$ in Q'_{CT} are also mapped to NDCG relevance grades in the same manner.

Since NDCG has been criticized in past research for not correlating well with user satisfaction [2, 21, 24], we also use a measure that directly evaluates the rank correlation between two lists. Unlike NDCG, this measure does not require relevance judgments; the ranking of the lists are compared directly. We still show our results with NDCG, since it is a well-known measure widely used in the information retrieval community. As the rank correlation metric, we use the M Measure [3] which penalizes differences in rankings depending on the rank position, similarly to the weighted generalized version of Kendall's Tau and Spearman's footrule presented in recent work by Kumar and Vassilvitskii [16]. The M Measure is defined as follows:

$$M(t, r)@k = 1 - \frac{M'(t, r)@k}{norm} \qquad (12)$$

where t and r are two ranked lists for which M yields 1 if t and r are identical at rank k, and 0 if they are completely disjoint at rank k. M' is defined as follows:

$$\begin{aligned} M'(t, r)@k = &\sum_{Z} \Big| \frac{1}{\text{rank}_t(i)} - \frac{1}{\text{rank}_r(i)} \Big| \qquad (13) \\ &+ \sum_{S} \Big(\frac{1}{\text{rank}_t(j)} - \frac{1}{k+1} \Big) \\ &+ \sum_{T} \Big(\frac{1}{\text{rank}_r(l)} - \frac{1}{k+1} \Big) \end{aligned}$$

where $i \in Z$ is the set of overlapping documents between t and r, $j \in S$ is the set of documents occurring in t but not in r, and $l \in T$ is the set of documents occurring in r but not in t. Note that a document being 'absent' in a list is assumed to have rank $k + 1$, which is denoted by the penalizing denominator $k + 1$ in S and T. The notation $\text{rank}_x(y)$ denotes the rank of document y in list x. The M Measure gives a higher weight to documents whose rankings are closer in high ranks of r and t, whereas the weight is lower with greater disagreements and lower rankings in the lists. The normalization constant in Equation 12 depends on k, referring to the case when Z is empty and all documents in r and t fall into the sets S and T.

In our evaluation, we observe $M(t, r)@k$ for t being the *truth list* – the clickthrough ranked list in decreasing order of click frequencies – and for r – the reranked list by one of the reranking methods.

5.2 Results

We first conduct the experiments on the unreduced 972 test queries to measure the performance of the baselines and all reranking models. For all models, the parameters α and κ (for the reranking models) and ρ (for BoosLuc) are set using the training data. For estimating the components in these runs, the models have full access to the training clickthrough data. Table 5 shows the results for the popular queries.

We can see that the baseline system Luc is already very good with NDCG@1 = 0.894 and M@1 = 0.682, increasing at lower ranks up until 0.925 for NDCG@20 and 0.705 for M@20. However, we can significantly improve on this by just boosting with clickthrough data, which is what the second baseline BoosLuc represents. We observe NDCG@1 = 1.0, and at the other ranks we have almost perfect ranking, which is also reflected in the M Measure rank correlation. As for the reranking models Sim, Sub, Syn and Merged – they also perform very similarly to BoosLuc with Syn (the synonym queries reranking model) showing the largest gain. Syn beats all other methods significantly with p-value < 0.04

Table 5: Comparing the reranking methods on 972 popular test queries. All bold numbers are statistically significant over all non-bold ones at same ranks with p-value < 0.04 using the paired two-sided t-test.

Average NDCG@rank	Luc	BoosLuc	Sim	Sub	Syn	Merged
1	0.894	**1.0**	**1.0**	0.999	**1.0**	0.999
2	0.89	0.997	0.996	0.996	**0.998**	0.995
5	0.907	0.992	0.988	0.99	**0.994**	0.987
10	0.914	0.99	0.984	0.987	**0.991**	0.984
20	0.925	0.989	0.984	0.986	**0.991**	0.984
Average M@rank	Luc	BoosLuc	Sim	Sub	Syn	Merged
1	0.682	0.983	0.978	0.972	**0.989**	0.971
2	0.678	0.971	0.964	0.962	**0.978**	0.957
5	0.699	0.948	0.933	0.938	**0.957**	0.93
10	0.707	0.928	0.908	0.916	**0.935**	0.906
20	0.705	0.90	0.878	0.888	**0.907**	0.876
Fixed Parameters	–	$\rho = 1000$	$\alpha = 0.5$ $\kappa = 20000$	$\alpha = 0.6$ $\kappa = 20000$	$\alpha = 0.6$ $\kappa = 20000$	$\alpha = 0.5$ $\kappa = 20000$

using the paired two-sided t-test for both measures. This experiment shows that with popular queries in a popular system when a substantial amount of clickthrough data is available, we can successfully boost a well-performing baseline such as the Lucene system with clickthrough data from the query only. For even better performance, the synonym queries reranking model can be used.

We now consider what happens if far fewer clicks are available on documents. We vary the number of clicks available for *all queries*, N, between 0 and 20,000 to simulate a sparse system that gradually progresses to a system with larger amounts of clickthrough data. Tables 6 and 7 and Figure 2 with the graphs show the simulated sparse query runs in a sparse system with little clickthrough data available *for all queries*. With higher N, the performance eventually becomes identical to what is reported in Table 5, since this corresponds to running the models with maximum N.

In Tables 6 and 7 we present all the reranking models in comparison to BoosLuc with only at most 1-50 clicks available for every query in the system. We notice a big, significant difference between BoosLuc and the reranking models at all ranks for these clicks, with BoosLuc catching up as the number of clicks increases. At NDCG@1 and M@1, all reranking models are almost perfect, whereas at lower ranks Sub and Merged perform best. This is also the case for 10 clicks, although Merged is not statistically significant over all methods at NDCG@5, but it is significantly better according to the M Measure. Sub clearly dominates for both 1 and 10 clicks for both measures. At 20 clicks, Sub is not the strongest method any more. The merged reranking model shows the best performance together with Sim at ranks 1 and 2. At 50 clicks we can also see that Merged dominates the results. This is not surprising, since Merged contains related queries from all models and it therefore has an advantage over the other methods in extreme cases, such as when no subset queries are available, or no co-click queries or synonyms could be found. It is surprising that Merged does not perform best in Table 5, where more clickthrough data is available. One reason might be that the scores of many documents are heavily boosted due to the large number of related queries – and that unfortunately many of those documents are non-relevant.

The parameters were fixed on training data individually for every different click setting for each of the methods. Note that the fixed parameter α is consistently higher in Tables 6 and 7 ($\alpha = 0.8, 0.9$) for most reranking models than it was in Table 5 (0.5, 0.6), which is the weight on the related queries reranking part in Equation 4. This is required for making up for the sparsity of clicks, whereas for more popular queries such a high emphasis on the information from the related queries is not required. κ is very sensitive for different reranking models and depends on the number of clicks available in a sparse system, although lower values are preferred. In the runs with the popular queries in Table 5 it is more stable with $\kappa = 20000$.

Figure 2 shows all the runs for the reranking models and BoosLuc (always the solid line) with NDCG and M Measure performance at the measured ranks 1, 10, and 20 for varying click counts. Each graph shows the results at a fixed NDCG or M Measure rank on the y axis with varying clicks on the x axis for the methods. In all the graphs, the BoosLuc curve starts off low and improves the more clicks (on the query itself for this baseline) are available. However, after 500 clicks the effectiveness has plateaued and no further gains occur. In fact, there is only modest value to clicks after the first 300. Therefore, the presented click range is between 0 and 800 or at most 2,000 for the graphs; the numbers do not change much afterwards and gradually progress into the results in Table 5.

As for the first two graphs with NDCG@1 and M@1 on the y axis, there is little noticeable difference between the reranking methods, except for the fact that they all outperform BoosLuc. A slight dominance of Sub is visible in the M@1 graph. At NDCG@10 and M@10, this becomes much clearer: the lines for Syn and Sim are noticeably below the lines for Sub and Merged. This fact becomes even more evident at NDCG@20 and M@20, with the gap between the lines growing larger. Another point is that between 0 and 100 clicks, in all the graphs the performance between the related query reranking models is very similar. With more than 100 clicks, the changes become noticeable.

Figure 2: Comparing different reranking methods and BoosLuc on 972 simulated sparse queries. The x axis shows the maximum number of clicks available, whereas the y axis is NDCG or M Measure @ different ranks.

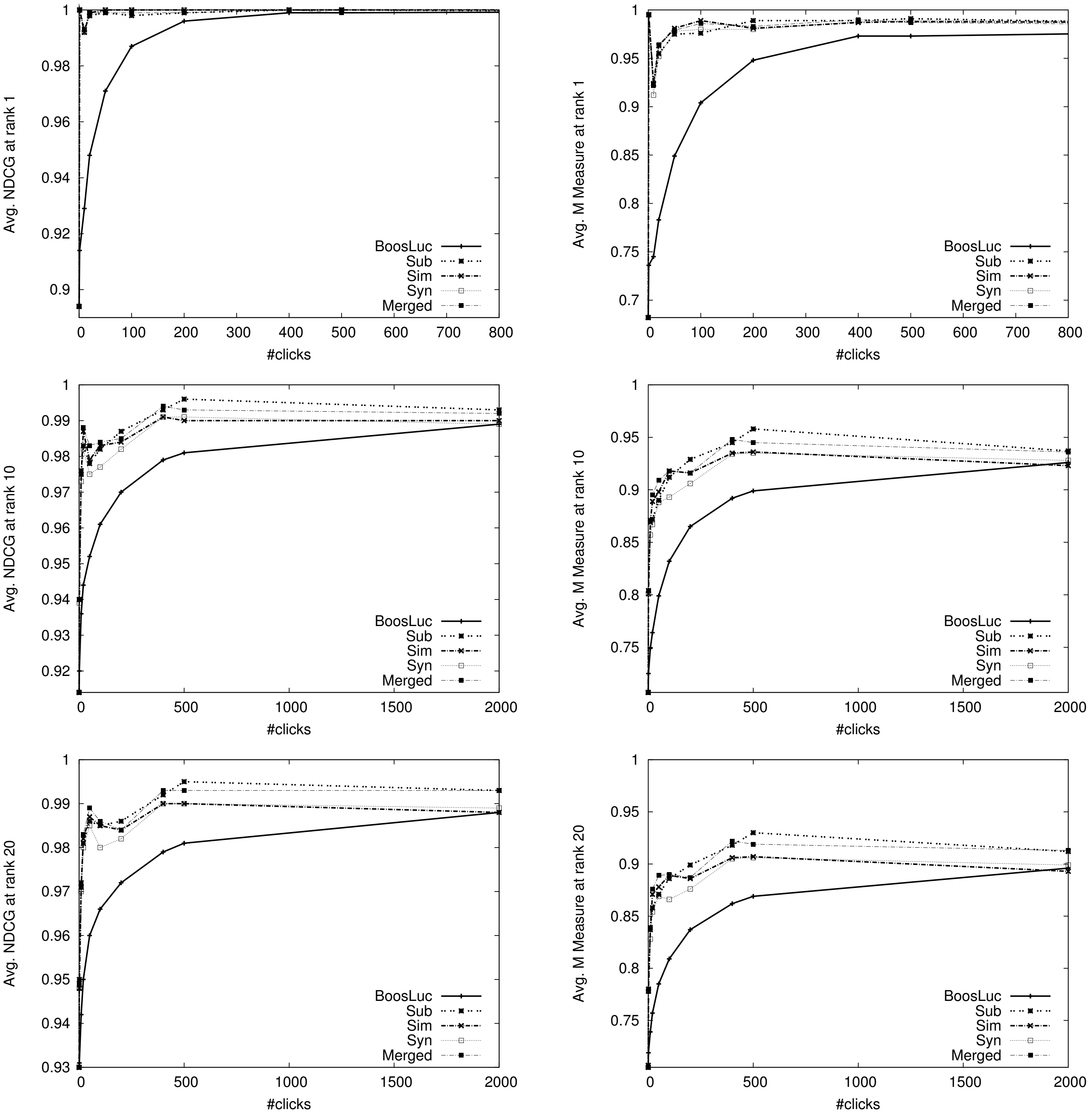

Table 6: NDCG results on comparing the reranking methods to BoosLuc on 972 simulated sparse queries when clickthrough data is sparse. The same results for ranks 1, 10, and 20 are shown in full in Figure 2. All bold numbers are statistically significant over all non-bold ones at same ranks and clicks with p-value $<$ 0.004 using the paired two-sided t-test.

clicks	$N = 1$					$N = 10$				
Average NDCG@ rank	BoosLuc	Sub	Merged	Sim	Syn	BoosLuc	Sub	Merged	Sim	Syn
1	0.914	**1.0**	**1.0**	**1.0**	**1.0**	0.929	**0.993**	**0.993**	0.992	0.992
2	0.903	**0.952**	**0.952**	0.949	0.949	0.919	**0.985**	**0.984**	0.982	0.977
5	0.915	**0.94**	**0.94**	0.939	0.939	0.932	**0.983**	0.982	0.981	0.979
10	0.92	**0.94**	**0.94**	0.939	0.939	0.936	**0.976**	**0.975**	0.975	0.973
20	0.931	**0.949**	**0.949**	0.948	0.948	0.942	**0.972**	**0.971**	0.971	0.97
Fixed Parameters	$\rho = 1000$	$\alpha = 0.8$ $\kappa = 1000$	$\alpha = 0.8$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.9$ $\kappa = 5000$	$\rho = 1000$	$\alpha = 0.9$ $\kappa = 10000$	$\alpha = 0.9$ $\kappa = 20000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.9$ $\kappa = 5000$
clicks	$N = 20$					$N = 50$				
Average NDCG@ rank	BoosLuc	Sub	Merged	Sim	Syn	BoosLuc	Sub	Merged	Sim	Syn
1	0.948	**0.998**	**0.999**	**0.999**	0.998	0.971	**0.999**	**0.999**	**1.0**	**1.0**
2	0.937	0.992	**0.995**	0.993	0.99	0.964	0.995	**0.997**	**0.997**	**0.997**
5	0.937	0.974	**0.983**	0.98	0.971	0.952	0.982	**0.987**	0.984	0.981
10	0.944	0.983	**0.988**	0.987	0.982	0.952	0.978	**0.983**	0.979	0.975
20	0.95	0.981	**0.983**	0.982	0.98	0.96	0.986	**0.989**	0.987	0.985
Fixed Parameters	$\rho = 1000$	$\alpha = 0.8$ $\kappa = 20000$	$\alpha = 0.8$ $\kappa = 1500$	$\alpha = 0.9$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 15000$	$\rho = 1000$	$\alpha = 0.5$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.8$ $\kappa = 1500$

Table 7: M Measure results on comparing the reranking methods to BoosLuc on 972 simulated sparse queries when clickthrough data is sparse. The same results for ranks 1, 10, and 20 are shown in full in Figure 2. All bold numbers are statistically significant over all non-bold ones at same ranks and clicks with p-value $<$ 0.02 using the paired two-sided t-test.

clicks	$N = 1$					$N = 10$				
Average M@ rank	BoosLuc	Sub	Merged	Sim	Syn	BoosLuc	Sub	Merged	Sim	Syn
1	0.736	**0.995**	**0.995**	**0.995**	**0.995**	0.745	**0.923**	**0.922**	**0.924**	0.912
2	0.715	**0.895**	**0.895**	0.891	0.891	0.73	**0.90**	**0.90**	0.899	0.878
5	0.724	**0.832**	**0.832**	0.828	0.828	0.742	**0.878**	**0.877**	0.876	0.86
10	0.725	**0.804**	**0.804**	0.801	0.801	0.749	**0.871**	**0.87**	0.869	0.857
20	0.719	**0.78**	**0.78**	0.778	0.778	0.739	**0.839**	**0.838**	0.837	0.828
Fixed Parameters	$\rho = 1000$	$\alpha = 0.8$ $\kappa = 1000$	$\alpha = 0.8$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.9$ $\kappa = 5000$	$\rho = 1000$	$\alpha = 0.9$ $\kappa = 10000$	$\alpha = 0.9$ $\kappa = 20000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.9$ $\kappa = 5000$
clicks	$N = 20$					$N = 50$				
Average M@ rank	BoosLuc	Sub	Merged	Sim	Syn	BoosLuc	Sub	Merged	Sim	Syn
1	0.783	0.955	**0.964**	**0.963**	0.952	0.849	**0.975**	**0.979**	**0.981**	**0.978**
2	0.769	0.933	**0.948**	**0.944**	0.926	0.836	0.957	**0.97**	**0.969**	**0.963**
5	0.764	0.886	**0.91**	0.902	0.88	0.814	0.92	**0.938**	0.929	0.921
10	0.764	0.872	**0.895**	0.889	0.867	0.799	0.89	**0.909**	0.898	0.888
20	0.757	0.858	**0.876**	0.871	0.854	0.785	0.871	**0.889**	0.878	0.869
Fixed Parameters	$\rho = 1000$	$\alpha = 0.8$ $\kappa = 20000$	$\alpha = 0.8$ $\kappa = 1500$	$\alpha = 0.9$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 15000$	$\rho = 1000$	$\alpha = 0.5$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 1000$	$\alpha = 0.9$ $\kappa = 5000$	$\alpha = 0.8$ $\kappa = 1500$

The results show very interesting trends: subset queries seem to be a more reliable clickthrough source for reranking than synonyms or co-click queries at lower clicks, which is an interesting observation. For best performance over a wider range of low clicks, the merged model together with the subset query model are most helpful, as we learned from Tables 6 and 7. When substantial clickthrough data is available, however, Syn outperforms all other methods as we saw in Table 5. Further, the results at rank 1 are near perfect for almost all the reranking models, whereas maintaining perfect reranking quality at lower ranks until 20 seems slightly more difficult for some of the methods. These results show the sensitivity of the methods to the number of clicks available and high-rank versus low-rank performance. What is evident in all the results however is that in a sparse system with sparse queries – even queries with up to 400 clicks – clickthrough boosting solely with the information from the original query is a poor choice, as is done with BoosLuc. Furthermore, using synonym queries can be helpful even when a system has large numbers of clicks available.

Finally, it is evident in Tables 6 and 7 and in Figure 2 that the two measures NDCG and M Measure mostly agree with one another with small exceptions, although the absolute improvement between BoosLuc and the reranking methods is greater with the M Measure.

6. CONCLUSIONS

In this paper, we tackled improving search results for sparse queries when the overall availability of clickthrough data is sparse. The aim was to approximate the ranking as defined by users' clicks. For this, we presented several reranking models that use clickthrough counts from related queries to better estimate the relevance of a document, given a query. The general reranking model estimates the relevance of a document by considering the quality of the related query and the likelihood that the related query will retrieve the given document. The models differ in their source of related queries, which may be similar queries sharing at least one co-click with the original query, subqueries, or synonym queries. Finally, a merged model combines the various sources of related queries into one as evidence for reranking documents.

The experimental results showed that with popular queries the synonym queries reranking model significantly outperforms basic clickthrough boosting of the original query, although all methods were able to greatly improve over the basic Lucene baseline often achieving nearly perfect results. For sparse queries in a simulated sparse system we observed that the related query reranking models are able to maintain the strong performance and significantly improve over the clickthrough boosted baseline with large gains. In a very sparse system with only 1-50 clicks available for every query, the subset query and merged reranking models performed best.

As future work, it would be interesting to conduct further experiments on other sparse web data sets or on other domain-specific query logs from small- or medium-sized systems to understand the generalizability and limitations of our models.

7. ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #IIS-0910884, and in part by UpToDate. We wish to thank UpToDate for access to its query log and retrieval system. We are particularly grateful to Jerry Greene's help in the time-consuming process of tagging data for our experiments. Any opinions, findings and conclusions or recommendations expressed in this material are the authors' and do not necessarily reflect those of the sponsors.

8. REFERENCES

[1] E. Agichtein, E. Brill, and S. Dumais. Improving web search ranking by incorporating user behavior information. In *Proceedings*, SIGIR '06, pages 19–26, New York, NY, USA, 2006. ACM.

[2] A. Al-Maskari, M. Sanderson, and P. Clough. The relationship between ir effectiveness measures and user satisfaction. In *Proceedings*, SIGIR '07, pages 773–774, New York, NY, USA, 2007. ACM.

[3] J. Bar-Ilan, M. Mat-Hassan, and M. Levene. Methods for comparing rankings of search engine results. *Comput. Netw.*, 50:1448–1463, July 2006.

[4] M. Bendersky and W. B. Croft. Discovering key concepts in verbose queries. In *Proceedings*, SIGIR '08, pages 491–498, New York, NY, USA, 2008. ACM.

[5] T. Brants and A. Franz. Google n-grams, Web 1T 5-gram Version 1, 2006.

[6] A. Broder, P. Ciccolo, E. Gabrilovich, V. Josifovski, D. Metzler, L. Riedel, and J. Yuan. Online expansion of rare queries for sponsored search. In *Proceedings*, WWW '09, pages 511–520, New York, NY, USA, 2009. ACM.

[7] comScore. comScore Releases March 2011 U.S. Search Engine Rankings. `http://www.comscore.com`.

[8] N. Craswell and M. Szummer. Random walks on the click graph. In *Proceedings*, SIGIR '07, pages 239–246, New York, NY, USA, 2007. ACM.

[9] Z. Dou, R. Song, X. Yuan, and J.-R. Wen. Are click-through data adequate for learning web search rankings? In *Proceedings*, CIKM '08, pages 73–82, New York, NY, USA, 2008. ACM.

[10] J. Gao, W. Yuan, X. Li, K. Deng, and J.-Y. Nie. Smoothing clickthrough data for web search ranking. In *Proceedings*, SIGIR '09, pages 355–362, New York, NY, USA, 2009. ACM.

[11] J. Huang and E. N. Efthimiadis. Analyzing and evaluating query reformulation strategies in web search logs. In *Proceedings*, CIKM '09, pages 77–86, New York, NY, USA, 2009. ACM.

[12] K. Järvelin and J. Kekäläinen. Cumulated Gain-based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.*, 20:422–446, October 2002.

[13] X.-M. Jiang, W.-G. Song, and H.-J. Zeng. Applying associative relationship on the clickthrough data to improve web search. In *Advances in Information Retrieval*, volume 3408 of *Lecture Notes in Computer Science*, pages 475–486. Springer Berlin / Heidelberg, 2005.

[14] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings*, SIGIR '05, pages 154–161, New York, NY, USA, 2005. ACM.

[15] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Trans. Inf. Syst.*, 25, April 2007.

[16] R. Kumar and S. Vassilvitskii. Generalized distances between rankings. In *Proceedings*, WWW '10, pages 571–580, New York, NY, USA, 2010. ACM.

[17] A. Mastora, M. Monopoli, and S. Kapidakis. Exploring query formulation and reformulation: A preliminary study to map usersÕ search behaviour. In *Research and Advanced Technology for Digital Libraries*, volume 5173 of *Lecture Notes in Computer Science*, pages 427–430. Springer Berlin / Heidelberg, 2008.

[18] J. Pickens, M. Cooper, and G. Golovchinsky. Reverted indexing for feedback and expansion. In *Proceedings*, CIKM '10, pages 1049–1058, New York, NY, USA, 2010. ACM.

[19] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings*, SIGIR '98, pages 275–281, New York, NY, USA, 1998. ACM.

[20] F. Radlinski and T. Joachims. Query chains: learning to rank from implicit feedback. In *Proceedings*, KDD '05, pages 239–248, New York, NY, USA, 2005. ACM.

[21] S. E. Robertson, E. Kanoulas, and E. Yilmaz. Extending Average Precision to Graded Relevance Judgments. In *Proceedings*, SIGIR '10, pages 603–610, New York, NY, USA, 2010. ACM.

[22] X. Wei, F. Peng, H. Tseng, Y. Lu, and B. Dumoulin. Context sensitive synonym discovery for web search queries. In *Proceedings*, CIKM '09, pages 1585–1588, New York, NY, USA, 2009. ACM.

[23] G.-R. Xue, H.-J. Zeng, Z. Chen, Y. Yu, W.-Y. Ma, W. Xi, and W. Fan. Optimizing web search using web click-through data. In *Proceedings*, CIKM '04, pages 118–126, New York, NY, USA, 2004. ACM.

[24] H. Zaragoza, B. B. Cambazoglu, and R. Baeza-Yates. Web search solved?: all result rankings the same? In *Proceedings*, CIKM '10, pages 529–538, New York, NY, USA, 2010. ACM.

[25] M. Zhao, H. Li, A. Ratnaparkhi, H.-W. Hon, and J. Wang. Adapting document ranking to users' preferences using click-through data. In *Information Retrieval Technology*, volume 4182 of *Lecture Notes in Computer Science*, pages 26–42. Springer Berlin / Heidelberg, 2006.