Robust Query Rewriting using Anchor Data

Nick Craswell
Bing, Microsoft
Bellevue, USA
nickcr@microsoft.com

Bodo Billerbeck
Bing, Microsoft
Melbourne, Aus
bodob@microsoft.com

Dennis Fetterly
Microsoft Research
Mountain View, USA
fetterly@microsoft.com

Marc Najork
Microsoft Research
Mountain View, USA
najork@microsoft.com

ABSTRACT

Query rewriting algorithms can be used as a form of query expansion, by combining the user's original query with automatically generated rewrites. Rewriting algorithms bring linguistic datasets to bear without the need for iterative relevance feedback, but most studies of rewriting have used proprietary datasets such as large-scale search logs. By contrast this paper uses readily available data, particularly ClueWeb09 link text with over 1.2 billion anchor phrases, to generate rewrites. To avoid overfitting, our initial analysis is performed using Million Query Track queries, leading us to identify three algorithms which perform well. We then test the algorithms on Web and newswire data. Results show good properties in terms of robustness and early precision.

Categories and Subject Descriptors

H.3.3 [**Information Retrieval**]: Retrieval Models

General Terms

Algorithms, Experimentation

Keywords

Query rewriting, anchor text

1. INTRODUCTION

Web search is typified by fast light-weight interaction. In a 2006 Web search log, the median query length was two words, the median click count was one and the median rank of clicked results was one [34]. These median values could hardly be more extreme, compared to other IR applications where significant time is spent on query reformulation and viewing multiple relevant results, for example [2]. Given the characteristics of Web search, it is natural that Web search evaluation focuses on achieving good early precision given short user queries, while also focusing on scaling to large collections [18].

Another characteristic of Web search that has attracted less attention from the research community is the importance of query response time. Web search results are typically returned in under one second, and experiments that introduce a delay of 100 to 400 milliseconds have demonstrated a significant impact on user engagement [24, 8], with users performing fewer queries and becoming less likely to click. This creates a balancing act. Improving early precision is positive for users, but if this increases response time by hundreds of milliseconds, the overall effect on users may be negative.

Some algorithms have been developed and published which translate, expand or adjust the query based on large-scale linguistic resources such as search logs and document sets, for example [12, 21, 16, 20]. Many of these rewriting methods can be run quickly, without requiring any extra search against a document corpus, so they are suitable for applications where response time is critical. They may also rewrite the query conservatively. An approach which adds fewer terms to the query can be executed more efficiently.

This paper follows that approach, of using linguistic resources to rewrite queries, with the goal of improving retrieval effectiveness. We follow the approach of Dang and Croft [13], using a large anchor graph as a linguistic resource for query rewriting. Despite its contribution in defining the anchor graph, that paper took a semi-manual approach, assuming the user can choose the most effective formulation from a set of automatically generated candidates. Here our contribution is to introduce new algorithms that achieve statistically significant improvements in retrieval effectiveness in a fully automatic setting. The new algorithms, which adopt the random walk approach of [11], show good effectiveness in early precision and robustness.

2. RELATED WORK

Query-document mismatch is a fundamental problem in information retrieval, with one solution being pseudo-relevance feedback (PRF), employing two rounds of retrieval. The first uses the original query, then the top-k results are mined for additional vocabulary terms. The second round uses the expanded query, which typically is a mixture of the original query terms and expansion query terms.

PRF can improve both precision and recall of web search [32]. However, given the constraints on response time mentioned in the introduction, we note that PRF introduces a significant extra delay to query processing. Query execution may be massively parallel, accessing many index servers and many document summarization servers [3]. However, certain parts of the process are sequential, with three major sequential phases being query expansion (if any), retrieval and summarization. These must be sequential, since retrieval can not commence until the query is finalized, and summarization can not commence until the top-k results have been identified.

To take expansion terms from the top-k documents, for example building a relevance model [23], their content, or some representation of their language model, must be processed. Computationally this may have comparable cost to the document summarization phase at the end of retrieval. Therefore, to apply PRF an extra round of retrieval and summarization is required during the initial query-expansion phase, and this must be complete before retrieval of the user's results can begin. This is even true for expansion against a reduced corpus such as Wikipedia [33, 15, 29]. For example, the Wikipedia-based expansion method of Elsas *et al.* [15] retrieves the top 1000 documents for the user's query from Wikipedia.

Instead of PRF, or in addition to it, query rewriting may be based on linguistic resources such as anchor text or query logs. These resources can potentially be accessed with $O(1)$ lookups or with efficient data structures such as a trie [7]. The rewritten query can be a spelling correction, a suggestion for the user, or a query that can be mixed with the user's query to improve retrieval effectiveness.

Jones *et al.* [21] take the user's query and rewrite it to form a modified query. Their application is an advertising scenario where recall is important and documents have few words. Papers on spelling correction [7, 16] assume that the user's query is ill-formed and attempt to correct it. Such models operate at the character level, whereas algorithms which introduce synonyms operate at the word level. Despite this difference, in other ways the algorithms are related, for example using a dataset of translation probabilities. In general they use linguistic datasets, for example a large-scale set of query reformulations from web searching users [21], the text from a set of 50 million web pages [16] or Wikipedia [30].

Cui *et al.* [12] carry out query expansion using a bipartite click graph, connecting a query node to a URL node if a click has been observed. They then extended the graph to encompass words in queries and words in documents. Craswell and Szummer [11] also use the click graph, performing a random walk to find a probability distribution over nodes from any given starting node.

Dang and Croft [13] induce a bipartite graph with similar properties to the click graph, based on anchor text from document hyperlinks. One advantage of this approach is that web crawls and anchor data are more freely available for academic and commercial purposes. For example, when building an enterprise search engine it is unlikely that there will ever be sufficient usage logs on that service to build good language and translation models for query rewriting. However, there is some chance that anchor text from the corpus or from external corpora can be used to build a query rewriting system.

While improved precision can be a good indication of the usefulness of a system, another – related – aspect is reliability: even if precision on average is greatly increased, the results of an expansion method may be perceived to be worse than that of a baseline system if for some queries precision is markedly reduced. Smeaton and Van Rijsbergen [27] found as early as 1982 that not all queries benefit from the same expansion method. More recently this problem has gained greater popularity amongst information retrievalists. For instance, Harman and Buckley [17] confirm Smeaton's and Van Rijsbergen's insight and a number of papers followed their workshop, such as this by Collins-Thompson [10], who defines query expansion as a risk/reward problem in a machine learning framework and achieves good robustness. Robust retrieval has also become a major focus of TREC [28] and remains a widely studied problem.

Kraft and Zien [22] find in a user study that sourcing expansion terms from anchor text using rank aggregation over the anchor frequency and lengths in word and characters is superior than sourcing those terms from the document collection. In this work, they find anchors for expansion that contain the full query as an exact match.

Amitay et al. [1] use queries instead of anchors to aid with retrieval. While acknowledging the practice to *stuff* a document with keywords gained from in-links, they instead pre-expand a document with queries that were used earlier in a user session that eventually ended with a user visiting the target page. They record modest improvements in precision measures, but decreased the number of query reformulations needed by users to find pages markedly. In a sense, this work inverts our approach by adding text to the target documents and thus avoiding query expansion all together. Similarly, [14] use session queries as implicit annotations, but also add explicit annotations as well as anchor text to target documents. They find that while difficult to obtain explicit annotations are very useful, implicit annotation can lead to an increase in early precision.

3. REWRITING ALGORITHMS ON THE ANCHOR GRAPH

An anchor graph $\mathcal{G} = (\mathcal{A}, \mathcal{U}, \mathcal{E})$ represents the anchor text descriptions in a hypertext corpus. It is a bipartite graph, connecting anchors $\mathcal{A}$ and URLs $\mathcal{U}$. An edge (a, u) exists in $\mathcal{E}$ if there are one or more links to URL u with anchor text a. This form of graph was introduced by Dang and Croft [13] and is analogous to the bipartite click graph, so we will begin with adapting click graph random walk algorithms [11].

Each edge in $\mathcal{E}$ has a weight, and we use those weights to calculate transition probabilities of the random walk. In the click graph work, the edge weight was a click count [12]. Here we weight (a, u) according to how often a was used to describe u. A problem is that a single author can easily create a large number of links, for example by putting the same link at the bottom of every page on a large site. Hence we count the number of unique linking hostnames that use a to describe u.

We generate our anchor graph $\mathcal{G}$ from the ClueWeb09 corpus, which has edges such as (*facebook*, `http://facebook.com`) and (*click here*, `http://trec.nist.gov`). In the cases where a ClueWeb09 page links to a document outside the corpus, we still use that u and (a, u) data, since we do not need any information about u other than its links from ClueWeb09 documents. Overall our graph has $|\mathcal{A}| = 1.26$ billion unique anchors and $|\mathcal{U}| = 4.82$ billion unique URLs.

```
Output of Q2Q algorithm:
q = new york department of labor

1.  P(new york state department of labor|q) = 0.115 [6,
0.08]
2.  P(nys department of labor|q) = 0.039 [6,0.19]
3.  P(department of labor|q) = 0.039 [5,0.01]
4.  P(new york|q) = 0.037 [8,0.00]
5.  P(new york labor department publications|q) = 0.018
[1,0.048]
...
208.  P(job source|q) = 4.98E-04 [1,0.02]
209.  P(employer information|q) = 4.98E-04 [1,0.01]
210.  P(http://w|q) = 4.98E-04 [1,0.00]
211.  P(employment resources|q) = 4.98E-04 [1,0.00]
212.  P(employment|q) = 4.98E-04 [1,0.00]
```

Figure 1: Example output of the Q2Q algorithm, for the query *new york department of labor*. The random walk probability is indicated, as well as [I,J] being URL set intersection size and Jaccard similarity.

We identify 27.4 billion links, which collapse to $|\mathcal{E}| = 5.96$ billion edges once we identify the distinct (a, u) pairs.

In the remainder of this section we introduce three anchor-driven query rewriting algorithms which we call query-to-query, phrase-to-phrase, and hybrid.

3.1 Query-to-Query Translation

Our query-to-query (Q2Q) algorithm first identifies the user's query in the anchor data $\mathcal{A}$, via an exact string match, after case folding and removing punctuation. If there is no match, then Q2Q produces no output. From the anchor node, the algorithm identifies all anchors that can be reached in two steps as candidate rewrite queries. For example, the query *new york department of labor* appears as an anchor in $\mathcal{A}$, and has 21 edges in $\mathcal{E}$. From the 21 URL nodes in $\mathcal{U}$ we reach 212 anchor nodes in $\mathcal{A}$.

Having identified all two-step candidates, we then calculate features that indicate a good rewrite. Our primary feature is the random walk probability [11] from a two-step forward random walk with no self-transition probability. As in that paper, the transition probability is calculated based on edge weights, with the probability of following an edge being the edge's weight divided by the total incident edge weights of the node. Exploring other edge weighting and random walk parameters is left for future work.

We calculate two additional Q2Q features based on the URLs adjacent to q and q'. Let U_q and $U_{q'}$ be the set of URL nodes adjacent to q and q' respectively. One feature is the number of URLs co-cited by q and q', which is the cardinality of the intersection $|U_q \cap U_{q'}|$. The other feature is the Jaccard similarity of the URL sets, which is $|U_q \cap U_{q'}|/|U_q \cup U_{q'}|$. The Jaccard similarity is related to the random walk probability, but is also affected by the degree of q'. If we were to increase $|U_{q'}|$ purely by connecting q' to URLs outside U_q, the random walk probability $P(q'|q)$ would be unaffected, but Jaccard similarity would decrease.

Figure 1 shows some examples of Q2Q feature values. The highest-scoring rewrites under random walk probability and Jaccard similarity are at positions 1 and 2 respectively, and seem strongly related to the original query. The highest-scoring rewrite on intersection size is *new york*, which misses the Department of Labor aspect of the original query. There are also a large number of rewrites with intersection size of 1 which tend to be of low quality. For the remainder of the paper we post-filter the Q2Q results to remove candidates with intersection size of only one or two, which greatly reduces the set of rewrites we need to consider.

Using the Million Query Track queries, we noticed two problematic patterns in the Q2Q rewrites. First, some rewrites simply shorten the query, for example *new york dmv* rewritten as *new york*. Second, some rewrites contain Web noise words, for example adding the word *site* or replacing the query with *click here*. Since only a small number of noise words showed up in top rewrites, we found we could build the list by hand with a small amount of manual effort: *free*, *wikipedia*, *www*, *click*, *here*, *com*, *org*, *site*, *website*, *more* and *link*. Under the assumption that removing words via query shortening or adding noise words are unlikely to be helpful, we removed all such rewrites from consideration. A more comprehensive analysis of noise words is left as future work.

To implement the Q2Q rewriting algorithm, the Q2Q rewrites can be calculated offline for each node in $\mathcal{A}$. Then online a single lookup in a hash table can be used to access rewrites of the user's query (if any). If we know ahead of time the number of rewrites that will be used, we need store only those in the table.

3.2 Phrase-to-Phrase Translation

The Q2Q algorithm relies on having a query which already appears in $\mathcal{A}$. We can consider a broader range of rewrites by translating parts of queries, as in [21]. Rather than adapting the machine learning approach of this paper to our setting, we attempt to identify algorithms with no parameters that perform well. Such algorithms are easier to implement and reproduce since no training data and parameter tuning are required. In a setting with sufficient training data, our algorithms can be used as features in a larger model.

Our algorithms P2P and HYB are somewhat related to noisy channel models [26, 7] and statistical machine translation. A noisy channel model over search queries assumes that the user has a well-formed query in mind, but accidentally typed a noisy version of that query. Given the noisy query *sigir acomodation* one part of the model tells us about likely translations, that *acomodation* is a likely mistake for a user who meant *accommodation*. The other part tells us the likelihood of the target query *sigir accommodation*. In the context of machine translation this is called the target language model.

Our application differs from spelling correction models [7, 16], in that we operate over correctly spelled TREC queries. Rather than suggesting a replacement of the user's query, we combine the original query with one or more rewrites, with the goal of improving retrieval effectiveness. Unlike spelling correction, where the suggested query is typically much more likely than the query the user typed, we find effectiveness can increase even if the rewrite is less likely than the original.

For example, given the query *rental cars*, our translation features suggest candidate translations for *rental* such as *hire* and *vacation*. The target model tells us that *rental cars* is the most likely formulation of the three, *hire cars* is an order of magnitude less likely, and *vacation cars* is extremely rare. We may then find that performing retrieval with a weighted combination of *rental cars* and *hire cars* can improve retrieval effectiveness. Allowing rewrites that are less likely than the original query is appropriate because we have not assumed that the original query was noisy.

For translation data we have two algorithms, called P2P and HYB. The P2P (phrase-to-phrase) model is a substitution model as in [21], taking the translations from the pairs used in the Q2Q data. For example, if the anchor *tool rental company* is within two steps of the anchor *tool hire company* (co-citing one or more URLs) this provides some phrase-to-phrase evidence that $p = rental$ and $p' = hire$ are related. We use an alignment that drops shared words from one or both ends of the queries (in this case *tool* and *company*), then if something was removed and the remainder is non-empty we have an alignment. The translation data is built over all Q2Q pairs, such that $P(hire|rental) = l/L$, where l counts how many times alignment produced (p, p') and L counts how many times p was aligned with anything $(p, *)$.

```
P2P translation data:
p = rental

1.  P(rentals|p) = 0.215
2.  P(hire|p) = 0.123
3.  P(rent|p) = 0.044
4.  P(for rent|p) = 0.012
5.  P(vacation|p) = 0.010
...
550.  P(bracelet|p) = 3.12E-04
551.  P(hub|p) = 3.12E-04
552.  P(talent|p) = 3.12E-04
553.  P(project|p) = 3.12E-04
554.  P(web site|p) = 3.12E-04
```

Figure 2: Example translation data from the Q2Q algorithm, for the phrase *rental*. The translation probabilities for the 554 translated phrases sum to 1, and are proportional to how many Q2Q pairs produced each translation.

The HYB (hybrid) algorithm is a hybrid between the other two. Q2Q translates the entire query using whole-anchor data (Figure 1). P2P translates part of the query using data from anchor substitutions (Figure 2). HYB translates part of the query using whole-anchor data (Figure 1). Its translation probability is simply the random walk probability from the Q2Q dataset. Using our ClueWeb09 anchor graph, the difference between P2P and HYB translations can be quite pronounced, as shown in the example data in Figure 2 and Figure 3.

To use the P2P and HYB algorithms we also need to consider target probability. This helps us choose whether *rental*

```
HYB translation data:
p = rental

1.  rentals - 0.022 [99]
2.  car rentals - 0.017 [70]
3.  calendar - 0.003 [7]
4.  luxury automobile rental - 0.002 [1]
5.  netflix - 0.001 [6]
...
4475.  company name - 9.43E-07 [1]
4476.  passages of lorem ipsum - 9.43E-07 [1]
4477.  pizhichilphoto gallery - 9.43E-07 [1]
4478.  media - 9.43E-07 [1]
4479.  booking - 9.43E-07 [1]
4480.  if you want to advertise...  - 9.43E-07 [1]
```

Figure 3: Example translation data from the HYB algorithm, for the one-word phrase *rental*. The random walk probability is indicated, as well as [I] the URL set intersection size.

```
Anchor histogram data:
anchors ending in the word cars

1.  used cars - 11426
2.  new cars - 5396
3.  find local rental cars - 2743
4.  hybrid cars - 1832
5.  rental cars - 1649
...
245461.  zymol products for my cars - 1
245462.  zz top bob seger cars - 1
245463.  zz tops cars - 1
245464.  zz+ used cars - 1
245465.  zzz cars - 1
```

Figure 4: Example anchor histogram data, for all anchors ending in the word *cars*. The frequency is the total incident edge weight of that node in $\mathcal{G}$.

should translate to *hire* or *vacation*, depending on the context *rental* appears in. For example *hire cars* is more likely than *vacation cars*, yet *florida vacation properties* is more likely than *florida hire properties*. We use two target models. One is an anchor frequency histogram, in fact we use the total incident edge weight for that node in $\mathcal{A}$. Matching the candidate rewrite against this histogram is sufficient to make the target likelihood decisions described above, see Figure 4 for example data. The other target model we use is a language model over a large set of Web data, made available via API by Microsoft Research[1], as described by Huang *et al.* [19]. Other target models could also be considered, such as one derived from the ClueWeb09 body text itself.

To implement P2P or HYB, we precompute translations for each phrase in the anchor graph. Then given a k-word query we do a hash table lookup for each phrase to find its translations, and a further lookup of the target probability of each translated query. We note that each of our candidates is produced by applying exactly one translation. Adding more translations can be achieved by incorporating multiple single-translation rewrites into the expanded search.

Using the Million Query Track queries, we noticed some translations to stopwords, so manually built a small list of cases to avoid: *about*, *an*, *and*, *are*, *as*, *at*, *be*, *but*, *by*, *com*, *for*, *from*, *how*, *if*, *in*, *is*, *it*, *of*, *on*, *or*, *that*, *the*, *this*, *to*, *was*, *what*, *when*, *where*, *which*, *who*, *will*, *with*, *would*, *www*, *a*, *i* and *org*. Building such a list requires a small amount of manual effort, a more comprehensive study of translation stopwords is left for future work.

4. ANALYSIS

The previous section introduced the rewriting algorithms of Q2Q, P2P and HYB, but each algorithm has multiple possible variants. This section explores various alternatives, and finalizes the choice of algorithms before our experimental evaluation.

Dataset

To avoid overfitting, our analysis uses a queryset that is disjoint from our test sets. We choose the 2009 Million Query (MQ) Track queries [9], minus the 50 shared with the Web Track. Each of the 637 queries has very few judged documents, so we do not believe the results here are as reliable as those of our main experiments. Our goal instead is to

[1]Microsoft Web N-gram Services - http://web-ngram.research.microsoft.com/

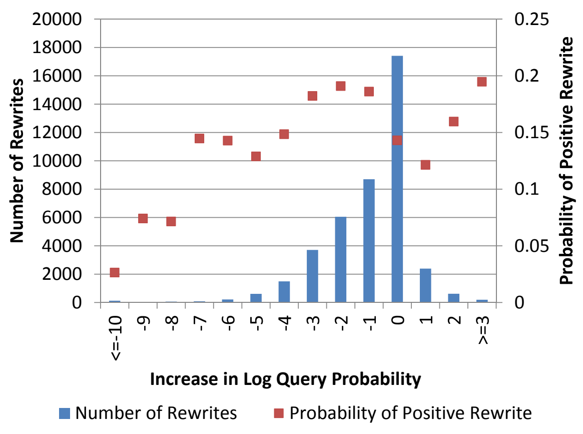

Figure 5: Differences in query log probability between rewrite and original, using Web language model data. Rewrites of higher frequency than the original are relatively rare, and do not seem to be significantly more likely to be positive according to our MQ judgments.

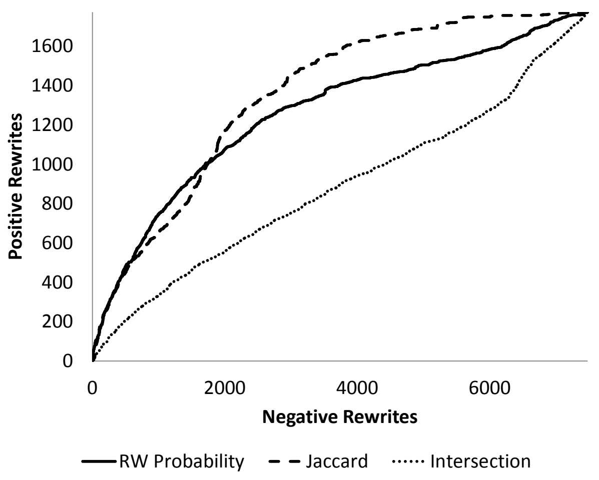

Figure 6: ROC curves analyzing all Q2Q rewrites, as indicated by the three different Q2Q features.

generate a very large number of rewrites for each query, and get a general view of which rewriting methods look promising. Performing this preliminary analysis allows us to make some decisions, and thus make fewer decisions on the test set(s).

The unfiltered output of the three algorithms for 637 queries is more than 1.3 million different query-rewrite pairs, the vast majority of which score low on all our features. To identify a more promising set of rewrites we apply the filters described in the previous section, and also remove all HYB and P2P rewrites with zero frequency in the anchor target model. These filters allow us to identify 41,639 rewrites, for which we gather target language model data via API. For each rewrite we also retrieve results from ClueWeb09 category A, ranking according to BM25F using title, body, anchor and URL fields.

Target Language Model

In introducing our algorithms we discussed the noisy channel model, which would tend to rewrite from a less frequent to a more frequent query. The rewrite should be more likely because the original query is noisy, and noise would tend to move from a likely query to a less likely one. We also considered an alternative hypothesis that the user's initial query is well formed, so it can be beneficial to rewrite to a less likely query, as in the *rental/hire cars* example.

To test this we identified a language model likelihood for each of our queries and rewrites. Then in Figure 5, the difference in likelihood between the rewrite and original query are the horizontal axis with one bin for each order of magnitude. We see that many rewrite candidates are as likely or less likely than the original query, which is not surprising. Using the MQ judgments, we then flag rewrites that had at least one relevant document in the top-1000 as *positive*, and plot the probability of drawing a positive rewrite from each bin. Results show that there are many cases where a less-frequent rewrite is positive. Generating the same graph using the ClueWeb09 anchor text histogram instead of language modeling probability showed a very similar pattern. We therefore will consider rewriting algorithms that have weak or no preference for more likely rewrites.

Q2Q Variants

The rewrites generated by Q2Q can be ordered by random walk probability, intersection size or Jaccard coefficient. To analyze these alternatives, we flag each Q2Q rewrite as positive if a document relevant to the original query occurs in the top-1000, and negative otherwise. Under this relatively weak definition of a positive rewrite, Q2Q produces 1,773 positive and 7,480 negative rewrites.

Figure 6 presents the results if we rank all Q2Q rewrites by each of the three potential features. The intersection signal is close to the diagonal, indicating there is little useful signal left after having used the feature as a threshold to cut down from our initial 1.3 million rewrites. The random walk probability and Jaccard similarity both seem like stronger signals, with random walk performing slightly better early on. The performance of the Jaccard similarity later in the ranking suggests that a combination of the two features could be a good alternative, but for simplicity we choose to use random walk scoring for Q2Q data in the remainder of this study.

P2P and HYB Variants

The previous analysis used all 41,639 query-rewrite pairs. We now consider several forms of our translation algorithms intended to choose the best rewrite for each query. We will also evaluate Q2Q as a reference point, using mean reciprocal rank (MRR) as our metric. Due to judgment sparsity many rewrites return results lists that are largely unjudged, so we chose MRR to give us an indication of where relevant documents are returned instead of focusing on top-20 precision or on recall.

Given the target language model analysis in Figure 5, we focus on comparing different target model variants. In one variant HYB0 there is no target language model. Given a query with w words, HYB0 finds all possible translations of n-grams in the query of length up to $w-1$, and applies the most likely one according only to the HYB (random walk) probability. HYB1 is the same, but rewrites which

Algo	microsoft home	my own background check
P2P0	ms home	my own criminal records
P2P1	ms home	personal background check
P2P2	windows home	my own business
P2P3	microsoft website	personal background check
HYB0	www.microsoft.com home	my h. david kotz named new inspector general at sec check
HYB1	microsoft corporation home	my own visit
HYB2	here home	my own background
HYB3	microsoft website	my own website
Q2Q	microsoft frontpage	–

Table 1: Rewrites for TREC Million Query Track queries 20591 and 20110. Most algorithms find good rewrites for *microsoft home*, while for query *my own background check* many rewrites do not seem useful.

have zero frequency in the anchor histogram are eliminated. HYB2 is like HYB1 but multiplies the HYB probability by the language model likelihood of the rewrite query. HYB3 is like HYB1, and comparable to HYB2, but instead multiplies the HYB probability by the anchor histogram probability of the rewrite query. If Figure 5 were incorrect, and we should prefer higher-frequency queries all other things being equal, then we would expect HYB2 or HYB3 to perform best.

The variants of P2P are the same as for HYB, with variants being: no filtering, filtering on existence in anchor, incorporating a language model probability and incorporating anchor histogram probability.

Example query rewrites are given in Table 1, showing the strongest rewrite for each algorithm. The examples show some typical patterns, such as the relatively sparse Q2Q having no output in one case, and HYB0 producing a nonsensical rewrite. Table 2 presents the MRR results. In this paper we present metrics multiplied by 100 for readability, so MRR of 1.1 is actually 0.011. The MRR results are low for this corpus due to the very small number of judgments per topic and the relatively large corpus. Unlike Figures 6 and 5 which considered thousands of rewrites, this table considers only the best rewrite for each query according to each ranking, so is potentially the most susceptible to noise. However, the results are consistent with Figure 5. The variants of types 2 and 3 perform not as well as the type 1 variants, for both HYB and P2P. The type 0 variants perform worst, indicating that some target model is necessary, but its most useful function is to filter out nonsense rewrites such as *vacation cars*.

The table also presents the coverage of each approach, since in some cases there are no possible rewrites. The 0 variants with no filter rewrite the most queries, although the filtered variants still have better coverage than Q2Q. Q2Q rewrites around a third of queries compared to the translation approaches, but the MRR on the – relatively few – rewritten queries is good.

5. RETRIEVAL EXPERIMENTS

Having finalized our rewriting algorithms, we test them on the TREC robust04 test collection (250 topics), the TREC GOV2 test collection (150 topics) and the ClueWeb09 adhoc test collection (150 topics). To set parameters in each case we use three-fold cross validation, choosing parameters on two folds and testing on the third, then present overall results for the three test folds.

Our unexpanded baseline is the default Indri language model (LM) ranking, and our expansion baseline is the Indri implementation of Relevance Model (RM) relevance feedback [23]. The latter model has multiple parameters, in our cross-validation we set three parameters: original query weight, number of PRF documents and number of terms. In all our Indri experiments we use Dirichlet smoothing ($\mu = 2500$).

	MRR	Queries	Affects	Data
P2P0	1.1	602	Phrase	P2P
P2P1	2.0	545	Phrase	P2P-AnchorFilter
P2P2	1.6	545	Phrase	P2P-LMProb
P2P3	1.9	545	Phrase	P2P-AnchorProb
HYB0	1.4	570	Phrase	Q2Q
HYB1	4.0	508	Phrase	Q2Q-AnchorFilter
HYB2	2.8	508	Phrase	Q2Q-LMProb
HYB3	2.6	508	Phrase	Q2Q-AnchorProb
Q2Q	1.1	179	Query	Q2Q

Table 2: MRR results on Million Query Track data. P2P1 and HYB1 perform best. Filtering for existence in anchor data (AnchorFilter) performs better than also multiplying by a target likelihood (LMProb or AnchorProb). Q2Q has lower MRR, though achieves this by rewriting only 179 queries.

		P@5	P@10	P@20	MAP	GMAP
LM	nostem	46.1	40.2	33.5	22.4	12.2
RM	nostem	*45.9*	40.6	34.5	26.5*†	12.3
QR	nostem	46.8	41.7*	34.4*	23.3*	13.0*
LM	stemmed	47.2	42.0	35.2	24.5	13.8
RM	stemmed	47.4	42.7	36.4*	28.1*†	*13.6*
QR	stemmed	47.4	42.8*	35.5	24.9*	14.1

Table 3: Overall results for robust04, with cross validation optimizing for MAP. Annotations: (*) Significantly better than LM with $p < 0.05$, (*italics*) numerically worse than LM and (†) significant difference between RM and QR with $p < 0.05$.

For comparison with RM we incorporate our three best rewriting algorithms (P2P1, HYB1, Q2Q) into algorithm QR, which has two parameters. One parameter r is the maximum number of rewrites to take from each algorithm. The other parameter w is the total weight assigned to rewrite queries. Our query formulation in Indri is a `#weight` combination of queries, each in its own `#combine`. The original query has weight 1.0, and each of k rewrites has weight w/k. Since there are three rewriting algorithms that each produce zero or more rewrites for a given input query, $k \leq 3r$.

In our cross validation setting, it would have been possible to tune both parameters separately for each of the three rewriting algorithms, and possibly use each algorithm's probability as a confidence score to further weight each rewrite. This would likely give a significant increase in effectiveness, but restricting ourselves to two parameters makes our approach more comparable with RM.

Overall results for robust04 are presented in Table 3, for an index with no stemming and also an index with Porter stemming. The cross validation optimization is towards mean average precision (MAP), and RM achieves significantly better MAP than QR and the language modeling (LM) baselines. However, it is numerically worse than the unexpanded baseline LM on some early precision measures and on geometric mean average precision (GMAP) [25]. The latter metric is intended to penalize a system for reducing

		P@5	P@10	P@20	MAP	GMAP
LM	nostem	46.1	40.2	33.5	22.4	12.2
RM	nostem	47.0	42.9*	35.5*	25.5*†	14.2*†
QR	nostem	47.1	42.2*	34.5*	23.5*	13.2*
LM	stemmed	47.2	42.0	35.2	24.5	13.8
RM	stemmed	47.7	43.7*	36.7*†	26.9*†	14.8*†
QR	stemmed	48.0	42.9*	35.4	24.9*	14.0

Table 4: Overall results for robust04 with cross validation optimizing for GMAP.

		P@5	P@10	P@20	MAP	GMAP
LM	nostem	56.8	55.5	52.6	27.0	19.0
RM	nostem	*54.9*	*54.5*	*51.3*	28.2*	*18.4*
QR	nostem	60.5*†	57.0†	55.0*†	28.7*	20.4*†
LM	stemmed	57.8	54.5	51.0	29.4	21.0
RM	stemmed	*55.8*	55.3	52.0	31.0*†	*20.5*
QR	stemmed	58.5†	56.1*	52.7*	29.9*	21.5

Table 5: Overall results for GOV2 with cross validation optimizing for MAP.

average precision (AP) towards zero, for example a small drop from AP=80 is penalized much less than a small drop from AP=1.

Given that RM had a numerical reduction in GMAP when optimizing for MAP, we conducted the same cross validation targeting GMAP itself. With that target, RM no longer has a loss in GMAP (Table 4) at the cost of losing about one point of MAP in absolute terms.

Overall results for GOV2 are presented in Table 5 for cross-validation optimizing for MAP and Table 6 for cross-validation optimizing for GMAP. The results in both cases are for an index with no stemming and also an index with Porter stemming. Cross validation with target MAP yields significant improvements over the baseline when stemming is not applied. However, RM again has a numerical (not significant) loss in GMAP, and QR is significantly better than RM on that metric. Note that the t-test for GMAP is conducted on log(AP) values.

When cross validation targets GMAP, QR reaches a very similar solution in terms of our target metrics, possibly because it is restricted to a single parameter (rewrite weight). The RM approach improves on GMAP but becomes worse on all other metrics than it was in MAP optimization. The next section analyzes this effect in more depth.

In Web collections, indexes typically allow unstemmed queries, for example so that a search for *david hawking* is not met with documents about a hawk or some hawks, or even Tennessee politician David Hawk. Thus for ClueWeb09 experiments we do not apply Porter stemming, and thus rely on RM and QR to introduce stem variants where appropriate. We also fix the number of rewrites for QR to 1, exploring a one-dimensional parameter space, instead of two-dimensional.

In our ClueWeb09 experiments, we did not build an Indri index, due to the size of category A English subset. Instead we ran the system from [6] and tested for significant gains due to rewriting over a baseline with unexpanded queries. Overall results are presented in Table 7, with cross validation optimized for NDCG@20. We found that optimizing for NDCG@20 yielded marginally better results in terms of statistical significance, but having done so ERR@20 was more likely to be the significant metric. Here we show just those

		P@5	P@10	P@20	MAP	GMAP
LM	nostem	56.8	55.5	52.6	27.0	19.0
RM	nostem	*55.7*	55.8	*52.6*	28.2*	19.2
QR	nostem	60.5*†	57.0	54.6*†	28.5*	20.2*
LM	stemmed	57.8	54.5	51.0	29.4	21.0
RM	stemmed	*56.5*	55.8*	52.2*	30.5*	21.2
QR	stemmed	58.1	55.6	52.5*	30.0*	21.5

Table 6: Overall results for GOV2 with cross validation optimizing for GMAP.

Rewrite	Variant	ERR@20	NDCG@20
	NONE	14.3	24.1
QR	HYB1	14.3	24.0
QR	P2P1	14.7*	24.4*
QR	Q2Q	14.7	24.3
QR	ALL (1 param)	14.9*	24.4
QR	ALL (3 param)	15.2*	24.5

Table 7: Overall results for ClueWeb09, with cross validation optimizing for NDCG@20. Significant improvements are for P2P1, for all three algorithms with a shared weight and for all three algorithms with three separate weights.

	Target	P@5	P@10	P@20	MAP	GMAP
RM	P@5	46.3	42.2	35.9	27.3	14.0
RM	P@10	45.9	43.1	36.4	26.6	14.7
RM	P@20	46.4	42.9	36.5	27.1	14.6
RM	MAP	47.4	42.7	36.4	28.1	13.6
RM	GMAP	47.7	43.7	36.7	26.9	14.8
Range:		1.8	1.5	0.8	1.5	1.2
QR	P@5	48.0	42.9	35.5	25.0	14.1
QR	P@10	47.9	43.3	35.6	25.1	14.1
QR	P@20	48.6	42.8	35.9	25.1	14.2
QR	MAP	47.4	42.8	35.5	24.9	14.1
QR	GMAP	48.0	42.9	35.4	24.9	14.0
Range:		1.2	0.5	0.5	0.2	0.2

Table 8: Optimizing for different target metrics on robust04, with stemming. RM has a broader range of possible outcomes.

results, targeting NDCG@20. Best results are obtained with rewrites from three algorithms, and a marginal numerical improvement from using one weight per algorithm rather than a shared weight as in our robust04 and GOV2 experiments.

5.1 Robustness and Parameter Settings

Our overall evaluation results indicated mixed results for RM, that were very positive on MAP but sometimes negative on early precision and GMAP. The cause of this, and a comparison to QR, can be seen in Figure 7. The impact of QR on AP is small, yet it achieves consistent gains across our metrics, many of which are statistically significant. The GOV2 results also indicated that RM parameters optimized for one metric may be sub-optimal on another. This section studies these properties of RM, and compares to QR, with some further analysis of parameter settings.

In our first analysis, we repeat the cross validation for each of our five effectiveness measures. The goal is not to observe a significant improvement, but to analyze the variability of

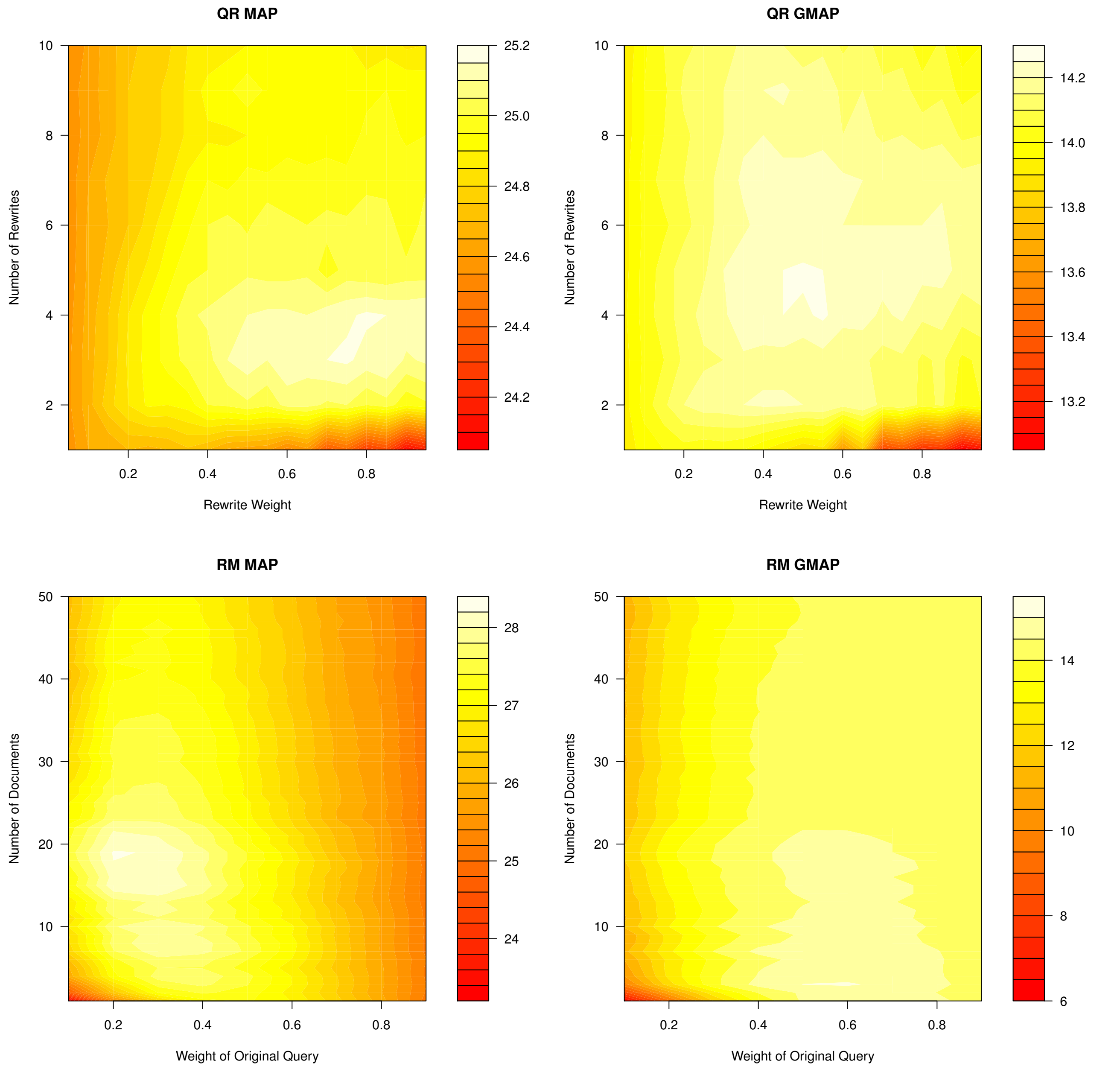

Figure 8: Parameter sensitivity for RM and QR on robust04, with stemming. Note that the plots for RM and QR can not be compared with each other, as they consider different parameters. Comparing the two QR plots, the optimum for MAP has good GMAP, and vice versa. For the RM plots the optima differ more, so there is a tradeoff between MAP and GMAP.

test performance under the different training targets. The experiment is related to experiments on varying training targets in Learning to Rank [31].

The variability of RM performance under different parameter settings is somewhat greater than the variability of QR (Table 8). This could be seen as a good property of RM, that it is powerful enough to be optimized for different metrics. To increase the power of QR approaches, perhaps more linguistic data could be incorporated, with exposure of more weighting parameters and a more sophisticated construction of the overall Indri query. RM's variability could also be seen as a disadvantage, since it has no parameter setting that is optimal for all metrics. For example, training on MAP gives the lowest GMAP, vice versa.

Comparing these results to the Learning to Rank results in [31], we see the same pattern, that our best results for a test metric are not always achieved when using the same metric for training. However, unlike that study QR can achieve good results when training on early precision.

Figure 8 presents an analysis of the difference between MAP and GMAP for the two algorithms. The figure indicates a parameter sweep rather than cross validation, though

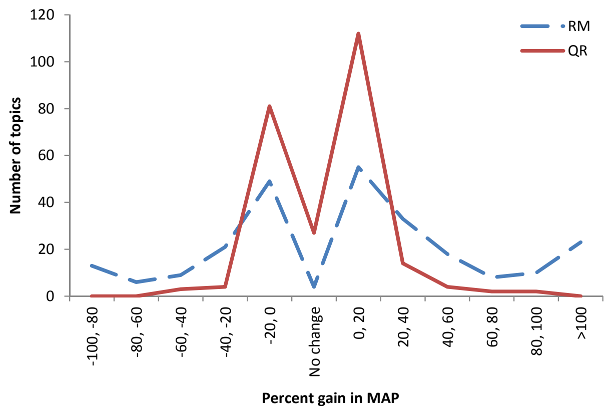

Figure 7: Robustness histogram for robust04 MAP, corresponding to Table 3 and with stemming. Our other experiments showed a similar pattern. RM has more topics at the extremes, including more with $> 100\%$ gain in MAP.

it uses the same full parameter grids as in our cross validation experiments. Note that due to the design of the algorithms the horizontal axes are reversed: QR has more effect as the rewrite weight increases (moving right), while RM has more effect when the original query weight decreases (moving left). Note also that the scales of the plots are different, and shown to the right in each case. Since RM has three parameters, for the purposes of this plot we hold the number of terms constant at its best value (20).

The stability of results for QR may be explained by the top two plots in Figure 8. The parameter settings for which MAP is optimal are similar to the parameter settings where GMAP is optimal. This is the likely explanation for the robustness of the algorithm. By contrast, the bottom two plots show the parameter settings for RM. If the goal is to optimize MAP, then the best parameter settings are with just under 20 documents and 0.4 weight on the original query, which means 0.6 weight on the relevance model. The parameter settings for optimal GMAP put much more weight on the original query, and may use fewer pseudo-relevant documents. This may explain the relatively poor GMAP performance of RM, since any choice of parameter settings is a tradeoff between MAP and GMAP.

Finally, we consider the parameter sensitivity of single rewriting algorithms on ClueWeb09, including rewriting algorithms that were evaluated in Table 2 on Million Query Track data. The results on adhoc Web Track data in Figure 9 indicate that HYB1, P2P1 and Q2Q were indeed the best algorithms, and are best applied at low weight.

6. CONCLUSIONS

We introduce several robust algorithms for query rewriting. Our general approach is to take a large linguistic dataset, in this case the ClueWeb09 anchor data, and use these to build translation and target models. We then apply these to generate rewrites of robust04, GOV2 and ClueWeb09 Web Track queries, and perform retrieval based on a mixture of original and rewrite queries.

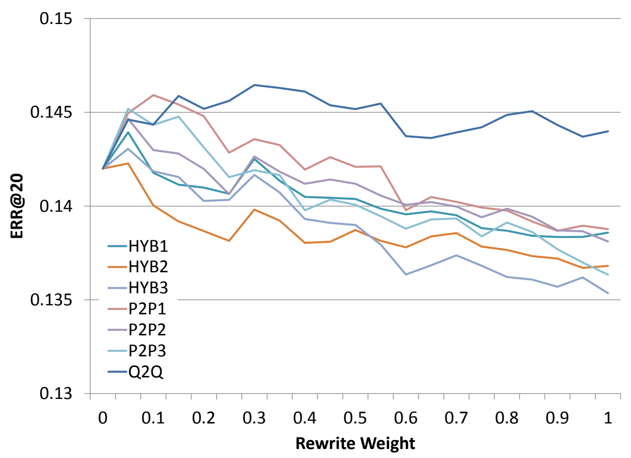

Figure 9: Parameter sensitivity for the best rewrite from each algorithm on ClueWeb09. Best ERR@20 is achieved at low weights for algorithms P2P1, HYB1, and Q2Q.

Having introduced the general form of three algorithms, we chose specific variants Q2Q, P2P1 and HYB1 based on analysis of tens of thousands of rewrites. The analysis indicated that a useful rewrite need not be more likely than the user's original query, consistent with the hypothesis that query rewriting can be useful even if the user's query was not "noisy".

Compared to a traditional PRF baseline, our QR algorithms do not have higher average precision, but have better early precision and are more robust. They yield significant improvements over an unexpanded baseline, and avoid losses in metrics that may be important to users such as early precision and geometric mean average precision. Under cross validation training, the QR approach yields consistent gains even if the target metric is changed, whereas the PRF baseline is much less stable, forcing a choice between higher MAP or higher GMAP.

Overall our results indicate our new QR approaches are competitive with traditional PRF without the need for two rounds of retrieval, making it a practical alternative for real-world systems where users care about early precision, response time and robustness. This can be achieved across collections, applying algorithms based on ClueWeb09 data to unrelated test collections robust04 and GOV2.

We note that approaches based on external linguistic resources can potentially be improved simply by adding more linguistic data. This suggests some promising follow-up work would be to identify more such resources, in particular resources that can provide within-language translations, in this case English-to-English. A further promising avenue would be to integrate such rewriting algorithms with query weighting approaches such as [4, 5]. With large-scale linguistic data for query rewriting, plus an integrated retrieval model, significant further improvements in retrieval effectiveness may be achievable.

7. ACKNOWLEDGMENTS

Thanks to Milad Shokouhi for his advice on good practices with Indri and RM parameter tuning and Dan Liebling for his help with obtaining the ClueWeb09 collection.

8. REFERENCES

[1] E. Amitay, A. Darlow, D. Konopnicki, and U. Weiss. Queries as anchors: selection by association. In *Proc. of the sixteenth ACM conference on Hypertext and hypermedia*, HYPERTEXT '05, pages 193–201, New York, NY, USA, 2005. ACM.

[2] J. R. Baron, D. D. Lewis, and D. W. Oard. TREC-2006 legal track overview. In *Proc. of TREC 2006*, 2006.

[3] L. Barroso, J. Dean, and U. Holzle. Web search for a planet: The Google cluster architecture. *Micro, IEEE*, 23(2):22 – 28, March-April 2003.

[4] M. Bendersky, D. Metzler, and W. B. Croft. Parameterized concept weighting in verbose queries. In *Proc. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 605–614, 2011.

[5] M. Bendersky, D. Metzler, and W. B. Croft. Effective query formulation with multiple information sources. In *Proc. of the 5th ACM International Conference on Web Search and Data Mining*, WSDM '12, pages 443–452, 2012.

[6] B. Billerbeck, N. Craswell, D. Fetterly, and M. Najork. Microsoft research at TREC 2011 web track. In *20th Text Retrieval Conference*, 2011.

[7] E. Brill and R. C. Moore. An improved error model for noisy channel spelling correction. In *Proc. of the 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 286–293, 2000.

[8] J. Brutlag. Speed matters for Google web search. Technical report, Google, 2009.

[9] B. Carterette, V. Pavlu, H. Fang, and E. Kanoulas. Million query track 2009 overview. In *Proc. of TREC 2009*, 2009.

[10] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proc. of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 837–846, 2009.

[11] N. Craswell and M. Szummer. Random walks on the click graph. In *Proc. of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 239–246, 2007.

[12] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma. Probabilistic query expansion using query logs. In *Proc. of the 11th International Conference on World Wide Web*, WWW '02, pages 325–332, 2002.

[13] V. Dang and B. W. Croft. Query reformulation using anchor text. In *Proc. of the 3rd ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 41–50, 2010.

[14] P. A. Dmitriev, N. Eiron, M. Fontoura, and E. Shekita. Using annotations in enterprise search. In *Proc. of the 15th international conference on World Wide Web*, WWW '06, pages 811–817, New York, NY, USA, 2006. ACM.

[15] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In *Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 347–354, 2008.

[16] J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. In *Proc. of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '08, pages 379–386, 2008.

[17] D. Harman and C. Buckley. The NRRC Reliable Information Access (RIA) workshop. In *Proc. of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 528–529, 2004.

[18] D. Hawking and N. Craswell. The very large collection and web tracks. In E. Voorhees and D. Harman, editors, *TREC: Experiment and Evaluation in Information Retrieval*. MIT Press, 2005.

[19] J. Huang, J. Gao, J. Miao, X. Li, K. Wang, F. Behr, and C. L. Giles. Exploring web scale language models for search query processing. In *Proc. of the 19th International Conference on World Wide Web*, WWW '10, pages 451–460, 2010.

[20] A. Jain, U. Ozertem, and E. Velipasaoglu. Synthesizing high utility suggestions for rare web search queries. In *Proc. of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 805–814, 2011.

[21] R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *Proc. of the 15th International Conference on World Wide Web*, WWW '06, pages 387–396, 2006.

[22] R. Kraft and J. Zien. Mining anchor text for query refinement. In *Proc. of the 13th international conference on World Wide Web*, WWW '04, pages 666–674, New York, NY, USA, 2004. ACM.

[23] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proc. of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, 2001.

[24] S. Levy. *In the Plex: How Google Thinks, Works, and Shapes Our Lives*. Simon & Schuster, 2011.

[25] S. Robertson. On GMAP – and other transformations. In *Proc. of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 78–83, 2006.

[26] C. Shannon, N. Petigara, and S. Seshasai. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 1948.

[27] A. F. Smeaton and C. J. Van Rijsbergen. The retrieval effects of query expansion on a feedback document retrieval system. *The Computer Journal*, 26(3):10–18, 1983.

[28] E. M. Voorhees. The TREC robust retrieval track. *SIGIR Forum*, 39:11–20, June 2005.

[29] Y. Xu, G. J. Jones, and B. Wang. Query dependent pseudo-relevance feedback based on Wikipedia. In *Proc. of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 59–66, 2009.

[30] X. Xue, W. B. Croft, and D. A. Smith. Modeling reformulation using passage analysis. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, CIKM '10, pages 1497–1500, New York, NY, USA, 2010. ACM.

[31] E. Yilmaz and S. Robertson. On the choice of effectiveness measures for learning to rank. *Inf. Retr.*, 13:271–290, June 2010.

[32] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proc. of the 12th International Conference on World Wide Web*, WWW '03, pages 11–18, 2003.

[33] W. Zhang and C. Yu. UIC at TREC 2006 blog track: a notebook paper. In *Proc. of TREC 2006*, 2006.

[34] Y. Zhang and A. Moffat. Some observations on user search behavior. In *Proc. of the 11th Australasian Document Computing Symposium*, pages 1–8, 2006.