Crowdsourcing for Robustness in Web Search

Yubin Kim
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA
yubink@cmu.edu

Kevyn Collins-Thompson
School of Information
University of Michigan
Ann Arbor, MI
kevynct@umich.edu

Jaime Teevan
Microsoft Research
Redmond, WA
teevan@microsoft.com

ABSTRACT

Search systems are typically evaluated by averaging an effectiveness measure over a set of queries. However, this method does not capture the the robustness of the retrieval approach, as measured by its variability across queries. Robustness can be a critical retrieval property, especially in settings such as commercial search engines that must build user trust and maintain brand quality. This paper investigates two ways of integrating crowdsourcing into web search in order to increase robustness. First, we use crowd workers in query expansion; votes by crowd workers are used to determine candidate expansion terms that have broad coverage and high relatedness to query terms mitigating the risky nature of query expansion. Second, crowd workers are used to filter the top ranks of a ranked list in order to remove non-relevant documents. We find that these methods increase robustness in search results. In addition, we discover that different evaluation measures lead to different optimal parameter settings when optimizing for robustness; precision-oriented metrics favor safer parameter settings while recall-oriented metrics can handle riskier configurations that improve average performance.

Keywords

crowdsourcing, slow search, risk-sensitive, robustness

1. INTRODUCTION

Traditionally, the effectiveness of search systems has been evaluated by computing an average performance measure over a set of queries. However, this may ignore critical differences in reliability if the improvements increase the variance of the performance measure. Some queries may perform much better at the expense of other queries that experience a significant decrease in performance compared to a baseline system (such as the search engine without the improvement). One of the reasons why academic research on query expansion has seen limited adoption in commercial systems is this increased risk. A commercial system cannot afford to deploy unstable results that may create a negative experience for a significant percentage of searches even if the technique improves average performance overall.

One potential way of avoiding serious failures, and thus reducing risk, in search results is to consult human judgment: humans are often better than machines at tasks like understanding complex natural language and relevance. Crowdsourcing services such as Amazon Mechanical Turk are a recent development that allow people to easily enlist the services of many crowd workers to complete what are typically small, quick tasks. In this paper, we integrate crowdsourcing into the process of search with the goal of using human understanding to introduce robustness into risky methods such as query expansion, and to improve search quality in general.

Crowdsourcing by its very nature is a slow process that cannot hope to achieve the sub-second response times typical of modern search engines. However, recently, researchers are beginning to explore a space called *slow search*, where search systems deliberately relax the stringent time constraints of modern search in order to deliver better results and user experience. Teevan et al. [14] found that users are sometimes willing to wait significant amounts of time if the end experience is much better.

Crowdsourcing has already been used in complex tasks such as question answering [9] and query understanding [7]. By introducing crowdsourcing into web search, we hope to leverage human intelligence to gain a better understanding of the query and its relationship to relevant documents. We look at two ways of incorporating crowdsourcing into search. First, crowd workers are integrated into the query expansion process where we use people to determine what terms are most related to the query. Second, we explore filtering the top of the ranked list using crowd workers to provide robustness by removing non-relevant documents that were retrieved in earlier stages of search.

The rest of the paper is organized as follows. In Section 2 related literature is reviewed. Section 3 introduces the details of the crowdsourced components and experiments to investigate their properties are presented in Section 4. Finally, Section 5 evaluates the runs that were chosen to be submitted to TREC.

	programming	computer	languages	object	computing	oriented	java
computer	2	9	0	0	6	0	0
programming	9	3	3	0	1	0	1

Table 1: Example of data collected for the query '*computer programming*'. Columns are candidate expansion terms and the numbers in the row indicate the number of workers who responded that the expansion term was related to the query term indicated by the first column of the row. Expansion terms are ranked based on query term coverage and query term relatedness; in this example, the top three terms are '*computer*', '*programming*', and '*computing*'.

2. RELATED WORK

We use crowdsourcing to increase the robustness of baseline query expansion techniques, and to filter an initial result list for more robust final result ranking. In this section, we first summarize the literature surrounding query expansion, and then discuss other ways crowdsourcing has been used in search.

There have been decades of research into query expansion. Pseudo-relevance feedback is one of the most popular forms of query expansion, using models such as Rocchio [13] and Lavrenko's relevance models [10] to calculate expansion terms. Although pseudo-relevance feedback often increases the average performance over a set of queries, it also typically increases the variance of query performance, which has helped restrict its use in real-world settings. Some efforts by researchers to address this issue include Collins-Thompson and Callan [4] and Crabtree et al. [6]. Both approaches used automated methods to increase query expansion robustness: Collins-Thompson and Callan achieved this through re-sampling while Crabtree targeted underrepresented query aspects discovered through issuing additional queries. In later work, Collins-Thompson [2] was able to significantly improve the robustness of existing query expansion methods by casting query expansion as a convex optimization problem over a graph of words, using a joint objective that maximized term relevance while minimizing term risk. In our work, human computation was used to increase robustness.

In interactive query expansion, researchers have investigated the usefulness of human feedback in query expansion with mixed results [12]. Diaz and Allan [8] explored the use of humans in selecting query expansion terms and found that human feedback can improve performance. Our research provides a stricter framework in which humans can contribute in an effort to better control the process. We also emphasize and analyze the gains in robustness rather than increases in average performance.

In broader uses of human elements in areas related to search, Demartini et al. [7] introduced CrowdQ, a system for understanding complex structured queries. Another related use of crowdsourcing for search-related tasks is presented by Bernstein et al. [1], where they explore a method of automatically generating short "answers" offline by using crowdsourcing for uncommon queries where curated answers may not be available. Jeong et al. have used crowdsourcing to build an automated question answering service for public questions Twitter [9]. Very recently a crowd-powered toolkit for search was described by Parameswaran et al. [11] Our approach differs from these in that it uses crowdsourcing within an existing algorithmic search framework.

3. METHOD

We experimented with two methods of utilizing crowdsourcing. The first introduced crowdsourcing into query expansion and used crowd judgments to select good expansion terms from an automatically generated candidate list. The second used the crowd to filter the final ranked list to remove non-relevant documents in the top ranks. With both methods, we used Microsoft's internal crowdsourcing platform, which draws workers from Clickworker.com.

3.1 Query Expansion

Typically, previous research has found that interactive query expansion (i.e., asking humans to pick expansion terms) does not improve average performance. People generally have difficulty in identifying terms with the best utility and often make sub-optimal decisions [12]. However, there is a lack of research in other benefits humans may bring to the process. Although improving upon the average performance of automated query expansion may be difficult, we hypothesized that using human intelligence to detect incongruous individual or collective choices of expansion terms, thus helping to avoid the worst expansion failures, would improve the robustness of query expansion.

Ruthven [12] found that simple term presentation interfaces are insufficient in soliciting good feedback from users. To combat this problem, we used a more structured approach to gathering user feedback. Rather than asking for terms related to a query as a whole, we solicited votes for expansion terms that are related to each individual query term. This procedure was informed by Collins-Thompson [3], who found that using expansion terms that covered more aspects of the original query reduced risk. Given the votes by the crowd workers, we selected expansion terms that have good coverage and strong correlation with query terms.

The procedure for our crowdsourced query expansion was as follows. For a query q (where individual query terms are denoted with q_i) a list of 10 candidate expansion terms $c = \{c_j : j \in \{1, \ldots, 10\}\}$ was generated using the Indri search engine's[1] built-in pseudo-relevance feedback algorithm. A recent snapshot of English Wikipedia was used as the expansion corpus.

Crowd workers were shown this list of candidate terms with a single term from the query q_i and the entire query q for context. They were each asked to select up to 3 expansion

[1]http://lemurproject.org/

	# of queries	Avg terms in query
Web 2012	50	2.32
Web 2013	50	3.34

Table 2: Summary of query statistics.

terms from c that were related to the highlighted query term q_i.

To ensure that the crowd workers understood the requirements of the task, only workers that passed a qualification test were allowed to complete the task. The qualification test was two manually created tasks with obvious answers; workers passed if their answers for both tests were correct.

The task for a single query term was completed by r_n crowd workers. An example of the final voting data can be seen in Table 1 for the query *'computer programming'* and seven candidate expansion terms.

From the results of the tasks, the probability $p(c_j|q)$ for each candidate term was calculated. By assuming the independence of query terms, $p(c_j|q) = \prod_i p(c_j|q_i)$, where

$$p(c_j|q_i) = \frac{v_{j,i}}{\sum_j v_{j,i}}$$

$v_{j,i}$ is the number of crowd workers who responded that c_j is related to q_i. We then re-ranked the candidate terms c_j by $p(c_j|q)$ and expanded the original query with the top r_k candidate terms using the query template:
`#weight(` w_o `#combine(` q `)` $(1 - w_o)$ `#combine(` $c_1 ... c_{r_k}$ `))`
In the example of Table 1, the top 3 selected expansion terms were '*computer*', '*programming*', and '*computing*'.

The number of workers r_n, the number of top candidate terms used r_k, and the weight of the original query w_o are adjustable parameters. In our experiments, we varied r_n between 1 and 10, r_k between 2 and 5, and w_o between 0.8 and 0.98. The weights of the individual expansion terms were set as the weights originally generated from the expansion corpus.

3.2 Result Filtering

In addition to query expansion, we also briefly experimented with using the crowd to perform result filtering. It was hypothesized that the crowd could be used as quality control to filter out poor results, leading to higher ranking robustness.

In this component, each document in the top f_k of the result list was judged by f_n crowd workers. If the majority of workers indicated that the result is non-relevant, it was simply removed from the ranked list. The end result is that relevant documents are moved up higher in the list. The numbers f_n and f_k are parameters that can be adjusted. In our experiments, we set $f_k = 10$ and varied f_n between 1 and 5 to explore the effect of additional workers.

4. EXPERIMENTS

The two query sets used to evaluate the method were the TREC Web Track queries from 2012 and 2013, which were created for the ClueWeb09[2] and ClueWeb12[3] corpora respectively. Table 2 presents a summary of the query sets. There are 50 queries in each set, with the queries being 2 to 4 terms long on average.

The corpora were searched using the Batch Query service provided by CMU[4]. The indexes in the service were built using the Indri search engine with the Indri default list of stopwords removed and the terms were stemmed using the Krovetz stemmer.

The baselines to which the results were judged against for robustness were released by the TREC Web Track. They were created from spam-filtering the corpora using the Waterloo spam scores [5] and searching them with Indri using its default pseudo-relevance feedback.

The indexes we used were not spam-filtered. Therefore, to approximate the baseline retrieval environment, spam documents were removed from the search results. In the cases of runs where the result filtering crowdsourced module was used, the spam filtering was done prior to the crowdsourced filtering.

The metric used is the official metric for the TREC Web Track in 2013, intent-aware expected reciprocal rank at 20 (ERR-IA@20). α is the risk-aversion parameter, where larger values indicate a larger penalty for losses and with larger α, negative values for the metric are possible, even for systems that perform better on average.

The 2012 query set was used to explore the range of the parameter settings and the 2013 query set was used to test the final results.

4.1 Effect of Parameter Settings

Table 3 presents the risk-sensitivity results of the crowdsourced query expansion method compared against the organizer-provided baseline. The table shows various settings for r_k, the number of terms used for query expansion, and w_o, the weight of the original query on the 2012 query set. α is the risk-aversion parameter.

Two trends can be discerned from the above data. First, an increase in the weight given to the original query increases robustness. This is unsurprising as the robustness is measured against the baseline formed from the original query and any risk introduced by the expansion term may be mitigated by relying more heavily on the original query.

The second trend is that using *more* expansion terms increase robustness. This may seem counterintuitive at first glance, but has a reasonable explanation. Including only a few terms is an all-or-nothing approach in that if none of them are good terms, the query will do poorly, but if both are good, then the query will get a large boost. By increasing the number of expansion terms, we can be more certain that at least one of them is a good term that can lead to an increase in effectiveness and result in a smaller, but more

[2]http://lemurproject.org/clueweb09/
[3]http://lemurproject.org/clueweb12/
[4]http://boston.lti.cs.cmu.edu/Services/

w_o	r_k	α = 0	α = 1	α = 5	α = 10
0.80	2	-0.02792	-0.07333	-0.25499	-0.48206
	3	-0.01375	-0.04732	-0.18159	-0.34942
	4	-0.01216	-0.04293	-0.16601	-0.31986
	5	-0.00358	-0.03075	-0.13944	-0.27529
0.90	2	-0.01647	-0.04914	-0.17981	-0.34314
	3	-0.01504	-0.04704	-0.17503	-0.33502
	4	-0.01086	-0.04039	-0.15852	-0.30617
	5	-0.01008	-0.03859	-0.15259	-0.2951
0.95	2	-0.00666	-0.03187	-0.13271	-0.25877
	3	-0.00922	-0.03562	-0.14125	-0.27328
	4	-0.00489	-0.02863	-0.12359	-0.24229
	5	-0.00487	-0.02861	-0.12357	-0.24226
0.98	2	-0.00585	-0.02995	-0.12637	-0.24689
	3	-0.00659	-0.031	-0.12865	-0.25071
	4	-0.00587	-0.02965	-0.12477	-0.24368
	5	-0.00556	-0.02942	-0.12488	-0.24421

Table 3: Effects of different parameter settings for original query weight w_o (0.8–0.98) and number of expansion terms, r_k (2–5) on the risk-adverseness of ERR-IA@20 for a range of α, the risk-aversion parameter, on the 2012 query set. Number of crowd workers used $r_n = 10$.

even boost across queries.

One may notice that the numbers for $\alpha = 0$ are negative, indicating that the crowdsourced query expansion did worse than the provided baseline on average. However, this is due to the differences in the indexing environments of our set up and that of the provided baseline; in Table 4, the *0 workers* run is the basic Indri pseudo-relevance feedback run and would be identical to the baseline if the index environment was the same, i.e., the metric would equal 0.0. However, due to the differences in indexing procedures, the metric is negative. When compared to an Indri pseudo-relevance feedback run (*0 workers*) and the run of the raw original query (*no exp*) from the same index enviornment set up, the crowdsourced method improves the average performance by a small amount ($\alpha = 0$).

Table 4 also explores the effect of adding more workers to the query expansion task. As expected, as the number of workers increase, robustness and average performance both increase because the additional opinions mitigate poor, outlier judgments. Because of the virtuous effect of adding additional crowd workers, we use $r_n = 10$ for all remaining experiments.

4.2 Effect of Evaluation Metrics

It has been long recognized in TREC that different systems perform best for different metrics. Corroborating this observation, we saw that the optimal parameter settings for risk-sensitivity were affected by the type of evaluation measure used. Table 5 summarizes our findings.

The columns in Table 5 are organized left to right from more precision-oriented to more recall-oriented metrics. When organized as such, the distribution of dark blue cells (which indicate the best parameter setting) create a diagonal pattern from the bottom left to the top right. This indicates that more precision-oriented metrics favor "safer" parameter settings and cannot tolerate risk, while recall-oriented metrics produce riskier parameter settings that can deliver larger gains.

workers	α = 0	α = 10
no exp	-0.00740	-0.25857
0	-0.00663	-0.25209
1	-0.00618	-0.24713
2	-0.00618	-0.24714
3	-0.00620	-0.24730
4	-0.00617	-0.24700
5	-0.00617	-0.24700
6	-0.00625	-0.24708
7	-0.00541	-0.24294
8	-0.00537	-0.24253
9	-0.00552	-0.24388
10	-0.00556	-0.24421

Table 4: Effects of increasing number of workers, r_n on ERR-IA@20 for $w_o = 0.98$ and $r_k = 5$. 0 workers indicate an unmodified Indri pseudo-relevance feedback run.

w_o	r_k	ERR-IA@10	P-IA@5	P-IA@20	MAP-IA
0.80	2	-0.48183	-0.44993	-0.28740	-0.10681
	3	-0.34680	-0.36653	-0.26670	-0.07670
	4	-0.31259	-0.28187	-0.25377	-0.07601
	5	-0.27000	-0.21653	-0.24887	**-0.07238**
0.90	2	-0.34530	-0.26720	-0.23230	-0.08848
	3	-0.34393	-0.33460	-0.21540	-0.07977
	4	-0.31026	-0.23527	-0.21058	-0.07835
	5	-0.30567	-0.23527	-0.20565	-0.07694
0.95	2	-0.26703	-0.24827	-0.21337	-0.08882
	3	-0.29215	-0.26980	**-0.20257**	-0.08516
	4	-0.24440	-0.20380	-0.20990	-0.08328
	5	-0.24487	-0.20700	-0.20965	-0.08244
0.98	2	-0.25056	**-0.19960**	-0.22343	-0.08780
	3	-0.25204	-0.23680	-0.22638	-0.08628
	4	**-0.24436**	-0.21600	-0.22297	-0.08560
	5	-0.24516	-0.21600	-0.22572	-0.08555

Table 5: Different metrics and their effects on the optimal parameter settings (in bold). $\alpha = 10$ but other values of α had similar effects.

This phenomenon is easily explained by the fact that recall-oriented metrics such as MAP consider a much larger set of documents than ERR. In ERR, because only the top few results contribute to the scores, the quality of every document matters and a single non-relevant result causes a large penalty. However, in MAP, the penalty of a single non-relevant result is reduced and thus a retrieval system can make riskier decisions when optimizing for this metric.

This further suggests that systems should use different robustness settings depending on the type of query and search needs of the user. A typical navigational query calls for

	α = 0		α = 10	
workers	ERR-IA@5	ERR-IA@20	ERR-IA@5	ERR-IA@20
0	-0.00452	-0.00556	-0.23578	-0.24421
1	-0.00948	-0.02444	-0.41351	-0.51000
2	-0.00332	-0.01719	-0.26101	-0.36647
3	0.01441	-0.00063	-0.23436	-0.32434
4	-0.00087	-0.01596	-0.26845	-0.37053
5	0.01332	-0.00135	-0.17084	-0.26977

Table 6: Crowdsourced result filtering combined with crowdsourced query expansion. Parameters for the query expansion component are $r_k = 5$, $r_n = 10$, $w_o = 0.98$. 0 workers is the run with crowdsourced query expansion, but without any result filtering.

methods with minimal risk, while a search engine may attempt riskier algorithms for a recall-oriented informational need as it is common in patent and medical search settings.

4.3 Effect of Result Filtering

The results of the crowdsourced result filtering are presented in Table 6, showing the effects of varying the number of crowd workers for two settings of $\alpha = \{0, 10\}$.

Overall, the result filtering component increased the robustness of the run. Its effects were especially pronounced at higher α values (higher risk-aversion) for ERR-IA@5. However, because filtering only affects the top 10 results, the increase is only present in metrics at smaller ranks, e.g., ERR-IA@5. In metrics at deeper ranks such as ERR-IA@20, the benefits of result filtering are no longer seen. In fact, the result filtering run does worse than the run without any filtering in $\alpha = 10$, indicating that some relevant results were removed during the filtering process. The reason for this is discussed further in Section 5.

Another item to note is that the result filtering run does very poorly with only a single crowd worker: much worse than the run without filtering. This is unsurprising as there was no quality control in this component; results only improve after having sufficient votes to mitigate the lower-quality votes.

Despite the large gains in robustness seen in ERR-IA@5 for high α values, we did not use the filtering component in any of our submitted runs due to its lackluster performance at deeper ranks.

5. SUBMITTED RUNS

From the above experiments, the following parameter settings were chosen for the 2013 query set and were submitted to TREC:

- msr_alpha0: $w_o = 0.8, r_k = 5$
- msr_alpha1: $w_o = 0.95, r_k = 5$
- msr_alpha5: $w_o = 0.98, r_k = 2$
- msr_alpha10: $w_o = 0.98, r_k = 5$

In addition, the following two runs were also submitted as runs for the adhoc task of the Web Track:

- msr_alpha0: $w_o = 0.8, r_k = 5$

run	α = 0	α = 1	α = 5	α = 10
msr_alpha0	-0.00004	-0.04189	-0.20929	-0.41854
msr_alpha1	0.01571	-0.01120	-0.11883	-0.25337
msr_alpha5	0.01101	-0.01748	-0.13144	-0.27388
msr_alpha10	0.01042	-0.01792	-0.13128	-0.27297
msr_alpha0_95_4	0.01561	-0.01225	-0.12370	-0.26300

Table 7: Results for submitted runs using ERR-IA@20.

	α = 0		α = 10	
workers	ERR-IA@5	ERR-IA@20	ERR-IA@5	ERR-IA@20
0	0.01059	0.01042	-0.28723	-0.27297
1	0.05708	0.04152	-0.41022	-0.49260
2	0.02997	0.01772	-0.26676	-0.33296
3	0.04384	0.03338	-0.22424	-0.24982
4	0.05665	0.04578	-0.23445	-0.26419
5	0.07301	0.06088	-0.21645	-0.23474

Table 8: Crowdsourced result filtering combined with crowdsourced query expansion. Parameters for the query expansion component are $r_k = 5$, $r_n = 10$, $w_o = 0.98$. 0 worker is the run with crowdsourced query expansion, but without any result filtering.

- msr_alpha0_95_4: $w_o = 0.95, r_k = 4$, another run that performed well

The results for the submitted runs are presented in Table 7, where ERR-IA@20 is reported for four values of $\alpha = 0, 1, 5, 10$. The best run for all α values was msr_alpha1, which was better than the participants' median for 17/50 queries for all α. In addition, the relative ordering of the runs are mostly the same for all ranges of α (msr_alpha5 and 10 switch ranks, but the differences are small). This indicates that the crowdsourced query expansion method was stable in robustness and the differences in accuracy are accounted by evenly distributed gains rather than from large variance.

As mentioned previously, the crowdsourced result filtering was not submitted as a run, but we choose to present the results for it in Table 8. The results for the 2013 query set differs from the results of the 2012 query set (Table 6). While we saw that the result filtering was not effective at deeper ranks for the 2012 query set, in the 2013 query set it gives a boost to robustness at both ERR-IA@5 and ERR-IA@20.

A possible reason for this may be due to the differences in the queries; the 2012 query set included many intentionally ambiguous queries for the diversity task such as '*kcs*'. This may have caused difficulties for the crowd workers to accurately judge relevance and relevant documents may have been removed from the ranked list as a result. The 2013 query set has fewer ambiguous queries and thus may have been easier to assess leading to more accurate judgments and increased robustness.

6. CONCLUSIONS

In this paper, two methods of integrating crowdsourcing into web search was discussed. The first method introduced

crowd workers into the query expansion process and used their judgments to select expansion terms that are strongly related to many query terms. The second used crowd workers to filter the top ranks of a ranked list to prevent non-relevant documents from being shown by collecting relevance judgments from the crowd.

We found that both methods increased robustness and that the crowdsourced query expansion produced stable results. However, the result filtering component was less effective in the 2012 query set which contained many ambiguous queries due to the difficulty in making relevance judgments.

When experimenting with parameters, it was found that increasing the number of crowd workers per task increased robustness in both methods and in addition, giving more weight to the original query and using more expansion terms increased robustness further in crowdsourced query expansion.

It was also observed that the optimal parameters for robustness of the crowdsourced query expansion were dependant on the retrieval metric used. In general, precision-oriented metrics preferred safer parameter settings while riskier parameter settings could be used in recall-oriented metrics. This observation leads to the implication that robustness should be situationally optimized depending on the information need of the user, based on whether it is precision-oriented (e.g., navigation queries) or recall-oriented (e.g., patent search).

7. REFERENCES

[1] M. S. Bernstein, J. Teevan, S. Dumais, D. Liebling, and E. Horvitz. Direct answers for search queries in the long tail. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pages 237–246, New York, NY, USA, 2012. ACM.

[2] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the Eighteenth International Conference on Information and Knowledge Management*, CIKM '09, pages 837–846, 2009.

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

[4] K. Collins-Thompson and J. Callan. Estimation and use of uncertainty in pseudo-relevance feedback. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '07, pages 303–310, New York, NY, USA, 2007. ACM.

[5] G. V. Cormack, M. D. Smucker, and C. L. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Information Retrieval*, 14(5):441–465, Oct. 2011.

[6] D. W. Crabtree, P. Andreae, and X. Gao. Exploiting underrepresented query aspects for automatic query expansion. In *Proceedings of the 13th ACM SIGKDD international conference on knowledge discovery and data mining*, KDD '07, pages 191–200, New York, NY, USA, 2007. ACM.

[7] G. Demartini, B. Trushkowsky, T. Kraska, M. J. Franklin, and U. C. Berkeley. CrowdQ : Crowdsourced Query Understanding. In *Conference on Innovative Data Systems Research (CIDR)*, 2013.

[8] F. Diaz and J. Allan. When less is more: Relevance feedback falls short and term expansion succeeds at hard 2005. In E. M. Voorhees and L. P. Buckland, editors, *TREC*, volume Special Publication 500-266. National Institute of Standards and Technology (NIST), 2005.

[9] J. Jeong, M. Morris, J. Teevan, and D. Liebling. A Crowd-powered Socially Embedded Search Engine. In *Proceedings of ICWSM 2013*, 2013.

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

[11] A. Parameswaran, M. H. Teh, H. Garcia-Molina, and J. Widom. Datasift: An expressive and accurate crowd-powered search toolkit. In *1st Conf. on Human Computation and Crowdsourcing (HCOMP) 2013*, 2013.

[12] I. Ruthven. Re-examining the potential effectiveness of interactive query expansion. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, SIGIR '03, pages 213–220, New York, NY, USA, 2003. ACM.

[13] G. Salton. *The SMART Retrieval System—Experiments in Automatic Document Processing*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1971.

[14] J. Teevan, K. Collins-Thompson, R. White, S. Dumais, and Y. Kim. Slow Search or: How Search Engines Can Learn to Stop Hurrying and Take Their Time. In *Proceedings of the 7th Annual Symposium on Human-Computer Interaction and Information Retrieval*, HCIR '13, New York, NY, USA, 2013. ACM.