Using the Crowd to Improve Search Result Ranking and the Search Experience

YUBIN KIM, Carnegie Mellon University
KEVYN COLLINS-THOMPSON, University of Michigan
JAIME TEEVAN, Microsoft Research

Despite technological advances, algorithmic search systems still have difficulty with complex or subtle information needs. For example, scenarios requiring deep semantic interpretation are a challenge for computers. People, on the other hand, are well suited to solving such problems. As a result, there is an opportunity for humans and computers to collaborate during the course of a search in a way that takes advantage of the unique abilities of each. While search tools that rely on human intervention will never be able to respond as quickly as current search engines do, recent research suggests that there are scenarios where a search engine could take more time if it resulted in a much better experience. This article explores how crowdsourcing can be used at query time to augment key stages of the search pipeline. We first explore the use of crowdsourcing to improve search result ranking. When the crowd is used to replace or augment traditional retrieval components such as query expansion and relevance scoring, we find that we can increase robustness against failure for query expansion and improve overall precision for results filtering. However, the gains that we observe are limited and unlikely to make up for the extra cost and time that the crowd requires. We then explore ways to incorporate the crowd into the search process that more drastically alter the overall experience. We find that using crowd workers to support rich query understanding and result processing appears to be a more worthwhile way to make use of the crowd during search. Our results confirm that crowdsourcing can positively impact the search experience but suggest that significant changes to the search process may be required for crowdsourcing to fulfill its potential in search systems.

CCS Concepts: • **Information systems** → **Crowdsourcing**; **Information retrieval query processing**; *Search interfaces*

Additional Key Words and Phrases: Slow search, crowdsourcing, information retrieval

ACM Reference Format:
Yubin Kim, Kevyn Collins-Thompson, and Jaime Teevan, 2016. Using the crowd to improve search result ranking and the search experience. ACM Trans. Intell. Syst. Technol. 7, 4, Article 50 (July 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2897368

1. INTRODUCTION

There are a number of aspects of the web search process that are particularly hard for search engines to address automatically. While computers can deal with large volumes of data and consider multiple alternatives, it is hard for them to algorithmically understand a person's information need or subtle meaning in a document. Tasks that require deep understanding are currently better suited for people. Recent research in human computation suggests that there is an opportunity for humans and computers to collaborate during search in a way that takes advantage of the unique contributions that

Authors' addresses: Y. Kim, Language Technologies Institute, Carnegie Mellon University; email: yubink@cmu.edu; K. Collins-Thompson, School of Information, University of Michigan; email: kevynct@umich.edu; J. Teevan, Microsoft Research, Redmond, WA; email: teevan@microsoft.com.

DOI: http://dx.doi.org/10.1145/2897368

each can provide. Via crowdsourcing, crowd workers can provide query and document understanding, while computers can provide fast, large-scale analysis.

While crowd-based approaches to complex tasks have worked well for everything from word processing [Bernstein et al. 2010a] to itinerary planning [Zhang et al. 2012], they are challenging to incorporate into web search because the crowd needs more time to process information than computers do. Despite advances with real-time crowdsourcing [Lasecki and Bigham 2014], search tools that rely on human intervention will never be able to respond to a query in a fraction of a second. However, recent research suggests that there are scenarios where a search engine could take significantly longer than this to return relevant content if the additional time were to provide a much better experience [Teevan et al. 2013; Teevan et al. 2014]. People often invest minutes, hours, or even days in complex or exploratory search tasks such as planning a vacation or researching a medical diagnosis. For over half of the time that people spend using a search engine, they are typically engaged in multi-query search sessions that take minutes or hours [Dumais 2013]. For important, difficult, or complex queries like these, crowdsourcing may be able to augment the search experience despite requiring additional time [Teevan et al. 2013].

In this article, we compare different strategies for incorporating crowdsourcing into the search process. After a discussion of related work (Section 2), we provide an overview of the different stages in the search process where a crowd worker's unique ability to understand text can be used. We argue that crowd workers are particularly valuable because they can extract meaning from a user's query and the documents identified as relevant to the query by the underlying search engine (Section 3). We show that crowd workers can be used to improve the search result ranking by replacing the components that address query and document understanding in a traditional information retrieval pipeline (Section 4). We find using people instead of algorithms for query expansion and result filtering reduces query variance to produce more stable results, but does not provide the large gains in retrieval effectiveness that would probably be necessary to make the use of crowdsourcing worthwhile, particularly for longer, more difficult queries. This prompts us to explore the use of crowdsourcing to enable new search experiences where searchers can express rich queries and receive effectively synthesized search results. Crowd workers help us understand rich entity queries by identifying and extracting attributes such as location and business hours from descriptive queries, and then provide rich result processing by producing detailed judgments on how much a candidate result matched the query on various attributes. The rich, structured results from this process enable us to create a new search experience for the users: we find that this exploration is very well received (Section 5). By studying several different ways to incorporate crowdsourcing into a search system, we provide insight into how search tools might create new and better search experiences, using human insight to compensate for their algorithmic and limitations.

2. RELATED WORK

Research in crowd-powered systems has explored how human computation can help solve information-retrieval-related problems. Most prior work in this area focused on using crowdsourcing to obtain human relevance judgments for evaluation [Alonso et al. 2008; Chen et al. 2013]. However, recent work has begun to look at how to use crowd workers to address users' information needs in various forms. Because of people's expectations around speed in search [Schurman and Brutlag 2009], the use of online crowds has primarily been explored in search contexts where people already expect long wait times. For example, socially embedded search engines monitor social networking sites and provide automated answers [Hecht et al. 2012; Jeong et al. 2013]. Other question-answering systems have used human users to help perform expert finding [Richardson

and White 2011] and market-based real-time services [Hsieh and Counts 2009]. However, stringent time constraints in search are not always necessary. Researchers are currently exploring a space called *slow search*, where quality is prioritized before speed for difficult queries where users are willing to wait [Teevan et al. 2013].

Our framework breaks down the task of retrieving documents related to a query into subtasks, each of which may be handled by automated or human-powered methods that include crowdsourcing. In a previous instance of this approach, Demartini et al. [2013] introduced a related approach with CrowdQ, a system for crowdsourced query understanding of complex structured queries. This work focused on query understanding and could easily fit into the framework presented in this article, as could other modules that take additional time to run but do not require human intervention [Crabtree 2007].

There are also several studies [Franklin et al. 2011, 2013; Marcus et al. 2011] that present systems that use the crowd to compute structured queries. In addition, researchers have built crowd systems to handle specific tasks such as image searching [Yan et al. 2010]. Bozzon et al. [2012] describe CrowdSearcher, a general platform for answering a class of queries such as ranking and clustering using social media networks. In our work, rather than building a social media platform for specific tasks, we more broadly explore different ways of using general crowd computation outside of a social media network.

The most closely related work to our explorations into crowd-based ranking was done by Parameswaran et al. [2013], who introduced an API for including crowdsourcing into search called DataSift. DataSift provides crowd-powered query reformulation and result filtering components, similar to what we explore in Section 4. Using 30 handcrafted queries, they conducted experiments on image and product corpora to demonstrate the efficacy of the approach. We complement and extend this work by using a set of 150 queries that combines queries from the Text REtrieval Conference (TREC) and queries extracted from commercial search logs. We also focus on text rather than images. Our work provides new insight into the effect of crowdsourcing on robustness: we find that, in contrast to the hand-crafted queries, the performance of difficult queries can be harmed by crowdsourcing if crowd workers are unsure of the queries' meaning. Furthermore, our work goes beyond the approaches explored by DataSift by experimenting with new search pipelines to explore new types of queries and deeper, rich result understanding with crowdsourcing (Section 5).

Several existing studies have explored approaches that use the crowd to change the existing search experience. For example, Law and Zhang [2011] discuss how to use crowd computing to take a high-level user goal (such as "I want be healthier") and identify individual subtasks, with the aim of finding web search queries whose results help accomplish the high-level goal. While this work uses the crowd to help users plan a sequence of queries, we use crowdsourcing to address the underlying web retrieval process for a single query. Another related use of crowdsourcing is presented by Bernstein et al. [2010b]. They explore a method of automatically generating inline answers using crowdsourcing for queries where curated answers are not available. The methods they use to extract short answers from documents bear some resemblance to the result understanding step in our experiments with entity queries. However, in Bernstein et al. [2010b], workers are required to actively extract text from a web document, whereas our task is a substantially easier one, only requiring a "yes, no, or maybe" answer. Furthermore, result understanding is only one step in the pipeline described in our work. Finally, unlike our approach, the method of Bernstein et al. [2010b] was conceived as an offline process requiring historical query log data.

In summary, the work presented in this article builds on existing approaches that incorporate crowdsourcing into the search process. While most prior work has used crowd workers for evaluation, some recent work has looked to replace the algorithmic

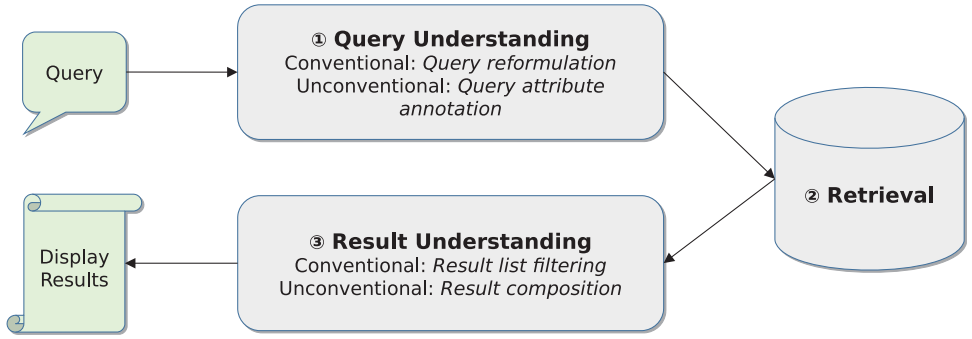


Fig. 1. Framework for including crowdsourced components into search pipeline.

components in search that are hard to automate [Parameswaran et al. 2013; Demartini et al. 2013]. We study how the crowd can be used to improve search result ranking on a larger dataset than has previously been explored, and this allows us to show that while the crowd can significantly increase robustness, it provides little overall benefit in terms of retrieval effectiveness. To make richer use of the crowd, we also explore several crowd-based approaches that change the existing search experience. While prior work in this space has focused on answering questions [Jeong et al. 2013] or queries [Bernstein et al. 2010b], we look at helping the search engine issue structured queries and tabulate the results.

3. CROWDSOURCING IN SEARCH

We explore the use of crowdsourced components at several different stages in the search process. A general framework for search can be seen in Figure 1. The three major stages in the framework are *query understanding*, *retrieval*, and *result understanding*. The query understanding stage is responsible for analyzing the user's expressed intent, such as via the query they enter into the search interface, to produce a query representation that the retrieval module can use. The retrieval stage uses this query representation to identify relevant documents. Then, in the result understanding stage, the result list returned from the retrieval stage is prepared for the consumption and understanding of the user.

While traditionally the modules that compose each stage have been fully automated, they could also be implemented using human computation. One possible way of incorporating human computation is to use crowdsourcing to seek incremental gains on existing automated techniques in a traditional ranked list interface. In this approach, the user experience is kept largely the same, while retrieval improvements are derived from better result ranking stems from the crowd's ability to understand natural language text more deeply than current machine algorithms.

Another approach is to attempt to find larger improvements by gathering deeper, more structured data with human computation and using this information to enable a significant improvement to the user experience of search, in addition to improvements to result ranking. While changing the user experience may be riskier, the combined benefits may be greater as well.

While many instantiations of the two broad philosophies are possible, we experiment with and compare one particular implementation of each of the approaches. Our focus is on improving search quality, particularly in the case of difficult queries, for which current search engines typically fail to deliver good results. The first approach uses human computation to replace two very common methods of improving search ranking quality: query expansion and result reranking. In the second approach, we narrow

our focus to long entity queries and use the crowd to enable a new search experience based on tabulated results. The following subsections detail the specific crowdsourcing techniques we explore for each stage of retrieval.

3.1. Query Understanding

The query understanding stage accommodates modules that change or annotate the query to better understand the user's intent. A classic example is query expansion, where related terms are added to the query to retrieve documents that may use different terminology. Other query understanding methods include key concept identification in queries [Bendersky and Croft 2008] and query reduction [Kumaran and Carvalho 2009]. Processes that annotate or classify queries also fit under the query understanding stage. Query classifiers may inform later retrieval processes so that specialized retrieval methods are used for different query classes [Kang and Kim 2003]. Query annotations may be used to enable searches in structured data such as tables [Sarkas et al. 2010].

We implemented two different ways of using human computation for query understanding. The first uses a module called *query reformulation* that aims to improve query expansion for ranking. Typically, queries are expanded using pseudo-relevance feedback, which expands the query using terms that co-occur with the query in the top-ranked documents from an initial retrieval ranking. This type of expansion involves both risk and reward: for some queries, adding additional terms can lead to significant improvements in retrieval effectiveness. In other cases, however, irrelevant terms can sometimes be accidentally added to the query [Collins-Thompson 2009], which can make results much worse. One method to automatically mitigate this risk and increased variance is to ensure the expansion terms are related to more than one of the terms in the original query [Collins-Thompson and Callan 2005], where relatedness is typically estimated through co-occurrence analysis or distance in a word relation graph. However, estimating the relatedness of terms can be a subtle semantic task that may be even more effectively done by humans. In our query reformulation component, we obtain judgments from crowd workers on whether candidate terms generated by an automated process are related to existing query terms.

The second way we use human computation for the query understanding stage is via a crowdsourcing approach that we call *query attribute annotation*. Descriptive entity queries such as "Italian restaurant in Seattle open for dinner on Sunday" specify multiple attributes such as location, cuisine type, and hours and tend to perform very poorly in existing search systems. The aim of this approach is to enable a larger improvement in the user experience of search through attribute identification and query segmentation of entity queries, a task where current computational methods fall short. Given the frequently ungrammatical structure of queries, where noun phrases are indiscriminately strung together, it can be a difficult task for a search system to extract coherent attributes from an entity query. The query attribute annotation component specifically addresses this shortcoming by instead asking crowd workers to perform segmentation during the query understanding stage. Furthermore, the component generates extra on-the-fly attributes by asking crowd workers to identify aspects of the query that were not adequately covered by a precompiled attribute list.

3.2. Retrieval

In the retrieval stage, the query representation from the query understanding stage is processed by a retrieval engine to generate a list of candidate search results. The data repository on which the search is conducted may be a traditional search index with inverted lists, or a structured repository such as a database or a knowledge graph. This stage of the search process is where machines excel over humans: machines are able

to sift through a large number of documents very quickly to pull out good candidates for additional processing, whereas humans would take an unreasonably long time to do the same task. Therefore, we rely on traditional algorithmic processes and do not introduce any crowdsourced modules to handle the retrieval stage.

3.3. Result Understanding

The final stage of the search process is result understanding. In this stage, the results produced from the retrieval stage are manipulated and processed to aid the user's understanding of the relevant content that has been identified. Examples of processes that fit this stage include result reranking (e.g., using learning to rank algorithms), snippet generation, and summarization. In addition, such modules may change how results are organized and presented, as is done in result clustering [Carpineto et al. 2009].

Because result understanding prepares the results for human interpretation, it can be valuable to add human computation into the loop to assist with assembling an effective presentation. One way of leveraging human computation is through a conservative method aimed at improving ranking but that does not change the search experience: *result list filtering*. This component simply filters out low-quality documents from a ranked list with crowdsourced relevance judgments.

However, it is also possible to use crowdsourcing to enable new search experiences. We explore a *result composition* component that synthesizes the results of entity queries into a tabular form. It asks the crowd workers to identify whether a candidate search result matches any of the attributes requested by the user in the query. With this information, the result composition module creates a table summarizing the results and how well they match the various query attributes, with the goal of helping users quickly identify good results.

4. CROWD-ENHANCED RANKING

Using these three stages as our guide, we begin by looking at conservative approaches to incorporate crowdsourcing into each stage in a way that preserves the overall user experience but improves the search result ranking. In this section, we describe the details of the ranking-centric components we implemented and introduce the dataset, queries, and metrics used to evaluate them. We then discuss the resulting successes and failures of focusing on ranking and show how this helps motivate our subsequent exploration into larger changes that impact the existing search experience.

4.1. Query Understanding: Crowd Reformulation

In this query understanding stage, we attempt to improve algorithmic query expansion using crowd workers. The initial query is first passed to an automated query expansion process. In this study, we used the Indri search engine's built-in pseudo-relevance feedback, which is based on Lavrenko's relevance model [Lavrenko and Croft 2001]. Candidate expansion terms were generated through pseudo-relevance feedback by running the queries against an English Wikipedia index and selecting the top 10 weighted terms from the vocabulary of the top 50 documents. The terms in the expanded query are then combined with those from the original query and passed to the crowdsourced query reformulation component for further processing by crowd workers.

The query reformulation component expects as input a query q, with individual query terms denoted by q_i, that is, $q = \{q_i : i \in \{1, \ldots, n\}\}$ that has been expanded by a list of candidate expansion terms $c = \{c_j : j \in \{1, \ldots, m\}\}$. For each term q_i in the query, the component creates a crowd task asking the worker to identify up to three candidate terms c_j that are most related to q_i, given the initial query q for context. Each of these tasks is given to r_n workers to complete.

Table I. Statistics of the Query Sets

	# of Queries	Avg Terms in Query
Difficult	100	6.28
TREC Web	50	3.34

With these task results, we can calculate the probability $p(c_j|q)$ for each candidate term. By assuming that query terms are independent, we have $p(c_j|q) = \prod_i p(c_j|q_i)$, where $p(c_j|q_i) = \frac{v_{j,i}}{\sum_j v_{j,i}}$. Here, $v_{j,i}$ is the number of crowd workers who responded that c_j is related to q_i. We then rerank the candidate terms c_j by $p(c_j|q)$ and reformulate the query so that it is expanded by the top r_k candidate terms. We do not modify the weights of the expansion terms provided by Indri.

The number of workers, r_n, and the number of top candidate terms used, r_k, are adjustable parameters. In our experiments, we used up to $r_n = 10$ workers and varied r_k from two through five terms.

4.2. Result Understanding: Crowd Filtering

In the result understanding stage, the result list filtering component takes as input a ranked list of documents. Then, for the top f_k documents, it collects relevance judgments from f_n crowd workers for each item.

We aggregate these judgments to obtain a relevance label for the (query, URL) pair by using majority voting, a simple consensus method for label aggregation that is widely used in information retrieval [Kazai et al. 2011]. With the majority vote label, if a majority of workers indicate that the result is nonrelevant, it is removed from the ranked list. The end result is that relevant documents are moved up higher in the list. While there exist a variety of more sophisticated aggregation methods (e.g., using EM [Ipeirotis et al. 2010]), these approaches are most suited to typical crowdsourcing scenarios where raters may be biased or unreliable. In our study, we used an internal corporate crowdsourcing service whose raters were experienced in relevance assessment, reducing the need to deal with label noise. We foresee using more advanced aggregation methods in future work when transitioning to use another, less reliable crowd platform.

The number of workers, f_n, and number of top-ranked documents used, f_k, are parameters that can be adjusted. In our experiments, we used the top $f_k = 10$ ranked documents and varied f_n from one through five to explore the effect of additional workers.

4.3. Experimental Setup

We evaluated how successfully the crowd could be used to improve search result ranking by looking at two different query sets: the queries published for the TREC 2013 Web Track and long, difficult queries extracted from the search log of a major search engine. These query sets and evaluation measures are described next.

4.3.1. TREC Web Queries. The first query set used to evaluate the system was published by the Text REtrieval Conference (TREC) for the Web Track in 2013, including relevance judgments made by trained assessors. We chose this query set for two main reasons. First, we wanted a publicly available dataset to help ensure the reproducibility of our results. Second, the 2013 Web Track introduced a risk-sensitive task that measured how well systems maximized gains while minimizing losses. We hypothesized that the crowdsourced components may be a safe method of improving queries, so we used the Web Track dataset to evaluate the system for robustness. The statistics of the query set are presented in Table I.

The corpus for these queries is the ClueWeb12 crawl.[1] For the retrieval stage, the TREC queries were submitted to the ClueWeb12 Batch Query Service maintained by Carnegie Mellon University,[2] which is an online service providing access to an index of the ClueWeb12 corpus built using the Indri search engine.[3] The index was stopped using the default Indri stop list and stemmed using the Krovetz stemmer. One thousand results were retrieved for each query for the initial result set and then spam-filtered postretrieval using the Waterloo spam scores [Cormack et al. 2011], similar the baseline run.

The baseline run was provided by the Web Track organizers and is generated using Indri with pseudo-relevance feedback over a spam-filtered collection of ClueWeb12.

4.3.2. Difficult Web Queries. We obtained difficult queries from a collection of internally identified queries with low retrieval effectiveness, extracted from several months of query log data from a commercial search engine. More specifically, query difficulty was determined by computing the normalized discounted cumulative gain (NDCG), based on editorial relevance judgments, of results for a set of "rare" low-frequency queries from the U.S. locale. To focus on deeper, more semantically oriented types of difficulty, we only considered low-effectiveness queries that did not have spelling errors, as determined by the search engine's automatic spell-checking and correction software: that is, where low effectiveness was not simply the result of poor matching caused by a mistyped or misspelled term. Each of these remaining queries was submitted to multiple commercial search engines, and NDCG@10 values computed for these ranked results. A query was identified as difficult if the maximum NDCG@10 effectiveness observed over all these search engines was less than a low-threshold value.

Although for privacy reasons we cannot release the actual queries obtained from commercial query logs, we can summarize some typical properties of difficult queries in this set, as well as provide examples inspired by actual queries, that help explain why search engines found them especially hard to satisfy. Summary statistics of the query set are presented in Table I.

Question/answer queries. A core subset of difficult queries had the form of longer, more natural language questions. For example, this included categories such as crossword puzzle clues, homework questions across a range of subjects, and users with certain symptoms seeking medical opinions. Users would also ask "why" questions with no readily available factual answer. These queries were typically seeking information that required deeper world knowledge, more sophisticated language understanding, and more advanced reasoning skills than currently available in commercial systems.

Malapropisms. Another category of difficulty involved malapropisms, which are cases where users wrote a correctly spelled but semantically incorrect or anomalous usage of a query term; for example, one query asked about "delta muscles" instead of "deltoid muscles." While commercial search systems have query alteration systems that are designed to detect these cases and offer a corrected query suggestion, such algorithms don't always succeed, leading to problematic results.

Multiattribute queries. A number of queries in the difficult set were seeking items or resources that needed to satisfy multiple attributes, sometimes with complex relationships, for example, a particular form of hardware ("cheap 3 inch wide pewter drawer pull") or some service or information involving a specific combination of attributes like department, location, demographic group, bureaucratic requirement, and so forth ("smith county tax forms for minority-owned businesses"). Commercial search engines

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

would return results satisfying some, but not all, of the desired properties—or with incorrect understanding of the relationships between the attributes. The restaurant queries we study in Section 5 fall into this category.

Procedural tasks. A number of difficult queries were seeking step-by-step instructions on how to do something common for a more unusual object or task ("how to replace battery in 1960s transistor radio") or how to do something unusual for a common item ("how to tarnish silverware"). In such cases, the majority intent for the common item or task would sometimes dominate the unusual intent or item in the ranking algorithm, leading to ineffective results.

Queries seeking very new information. In some cases, users were seeking new information that was so recent that commercial engines had not yet been able to crawl and index the relevant content, for example, from a rapidly developing news story or lyrics to a just-released song.

We used these difficult web queries to evaluate the hypothesis that crowdsourcing can improve search quality for unusually hard retrieval scenarios. We first created a baseline retrieval run by submitting the raw queries to a major search engine, retrieving the top 10 results as the initial candidate set.

To create relevance judgments for these results, we followed the TREC assessment process as closely as possible, to create gold-standard judgments comparable to those published by TREC. We also wanted to avoid relying on crowdsourced relevance judgments to evaluate a crowdsourcing system. Therefore, three contractors from an online freelancer marketplace called oDesk[4] were hired to provide relevance judgments for the difficult queries. The contractors were hired from within the United States and were selected through an interview process.

Individuals were provided with detailed instructions based on the training documents provided to TREC Web Track relevance assessors and were asked to judge results for a sample query. Three contractors with high accuracy were hired. These contractors judged a total of 3,501 query-URL pairs, and 42 URLs from one query were judged by all three contractors to calculate interannotator agreement. Note that the interview and training process used in hiring the assessors clearly distinguishes this source of relevance judgments from crowdsourcing, in which there are no guarantees as to what kind of or how many individuals will complete the given tasks, and is expected to yield more reliable relevance judgments.

Although the contractors made graded TREC-style judgments, due to low annotator agreement, their judgments were collapsed to binary (relevant vs. nonrelevant) judgments. The Cohen's kappa scores for the collapsed judgments were 0.64, 0.37, and 0.42 for the contractor pairs (1,2), (2,3), and (1,3), respectively. These somewhat low agreement scores may have occurred for two reasons. First, because the queries were mined from a query log, the contractors were not supplied with the topic descriptions that traditionally accompany TREC queries to provide more detail about the user's intent. Second, these queries were unusually long and difficult, so that even human assessors may have had trouble interpreting them and determining relevance.

4.3.3. Evaluation Methods. We evaluated effectiveness with the same evaluation measures used by the TREC Web Track: intent-aware expected reciprocal rank (ERR-IA@5) [Chapelle et al. 2009] and intent-aware precision at 5 (P-IA@5). The intent-aware version of metrics computes the metric for each possible intent of the query separately, then computes an average across the intents of the query. For example, the P-IA@k can be calculated as follows. Assuming M queries, let $N_t, 1 \leq t \leq M$ be the number of intents associated with query t. $j_t(i, j) = 1$ if the document for query t at rank j

[4]http://odesk.com.

is relevant to the intent i of query t. Otherwise, $j_t(i, j) = 0$. Then, the intent-aware precision at k is

$$\text{P-IA@k} = \frac{1}{M}\sum_{t=1}^{M}\frac{1}{N_t}\sum_{i=1}^{N_t}\frac{1}{k}\sum_{j=1}^{k} j_t(i, j).$$

For the difficult web query set, there is only one "default aspect" and the metrics thus reduce to regular ERR@5 and P@5. We also computed risk-sensitive versions of the metrics, introduced in the TREC 2013 Web Track.[5]

The risk-sensitive version of metric R for a run containing a set Q of n queries is defined as follows. Let Q_+ be the set of queries in Q where the method improves upon the baseline and Q_- be the set of queries in Q where the method does worse than the baseline. We define $\delta(q) = R(q_{\text{run}}) - R(q_{\text{baseline}})$ to be the difference between baseline and run effectiveness, giving the risk-sensitive measure R as

$$R_{\text{risk}}(Q) = \frac{1}{n}\left(\sum_{q\in Q_+}\delta(q) + (\alpha+1)\sum_{q\in Q_-}\delta(q)\right),$$

where α is a risk-aversion parameter. Maximizing R_{risk} when $\alpha = 0$ corresponds to maximizing average effectiveness without a risk component, while if $\alpha >> 0$, then query results in which run effectiveness is worse than the baseline effectiveness are penalized more heavily.

To guard the privacy of users who issued queries in the difficult query set, our experiments used an internal crowdsourcing platform that was not publicly accessible. Thus, the cost and speed of our experiments were not representative of typical crowdsourcing environments, and we defer the discussion of effective cost and time tradeoffs to a future study. Instead, our evaluation focuses on the effectiveness and user experience of our system. (A thorough discussion of cost and its relation to task completion speed for crowdsourcing tasks can be found in Mason and Watts [2009].)

4.4. Experimental Results: Crowd-Enhanced Ranking

The results using our crowd-based retrieval approach are given next. We show that while the gains in average effectiveness from using crowdsourcing to improve ranking were limited, the results also supported our hypothesis that crowdsourcing can reduce the *variance* in effectiveness that results from using risky operations like query expansion, increasing the reliability of the search system.

4.4.1. Web Track Queries. Table II shows the retrieval effectiveness and robustness measures computed for the 2013 TREC Web Track query runs using various parameter settings. Recall that r_k is the number of expansion terms added in the query reformulation module, r_n is the number of crowd workers per task (these runs do not include result filtering), and f_n is the number of crowd workers that judged each query-result pair. When applying the expansion terms with Indri queries, we explored query interpolation weights for the initial query in the range 0.80 to 0.98. The best results were consistently obtained with interpolation of 0.98, so we used this setting. Varying the number of expansion terms gave mixed results: using $r_k = 2$ had slightly better P-IA@5, but $r_k = 5$ terms gave better ERR-IA@5, the primary metric used in the TREC Web Track, and so we focus on those results here.

To evaluate the crowdsourced runs for robustness, the new risk-sensitive version of ERR-IA for TREC 2013 was used, which is computed with respect to a given baseline

[5]http://research.microsoft.com/trec-web-2013.

Table II. Effectiveness of Web Track Runs for Various Parameter Settings

run		ERR-IA@5 $\alpha = 0$	ERR-IA@5 $\alpha = 10$	P-IA@5 $\alpha = 0$
baseline		0.315	0	0.262
$r_k = 2$	$f_n = 5$	0.384†	−0.198	0.342†
$r_k = 5$	Indri	0.330	−0.265	0.290
	$r_n = 1$	0.327	−0.289	0.288
	$r_n = 5$	0.326	−0.287	0.291
	$r_n = 10$	0.326	−0.287	0.291
	$f_n = 1$	0.372	−0.410	0.323*
	$f_n = 5$	0.388†	−0.216	0.336†

"Indri" is the initial pseudo-relevance feedback run with no crowdsourcing. † indicates a statistically significant difference between a run and the baseline using the two-tailed paired t-test at the $p < 0.05$ level. * indicates significance at $p < 0.1$.

system (in our study, the baseline Indri results). For a given query, a system is penalized if its effectiveness is lower than that of the baseline system on the same query. The magnitude of this penalty is controlled by setting the α parameter: higher values for α give higher penalties when a system has lower effectiveness (ERR-IA) than the baseline effectiveness. For this study, we chose the conservative, risk-averse setting of $\alpha = 10$ (the ERR-IA@5 $\alpha = 10$ column of Table II). (Although only results for $\alpha = 10$ are reported, the trends for $\alpha = 1, 5$ were similar.)

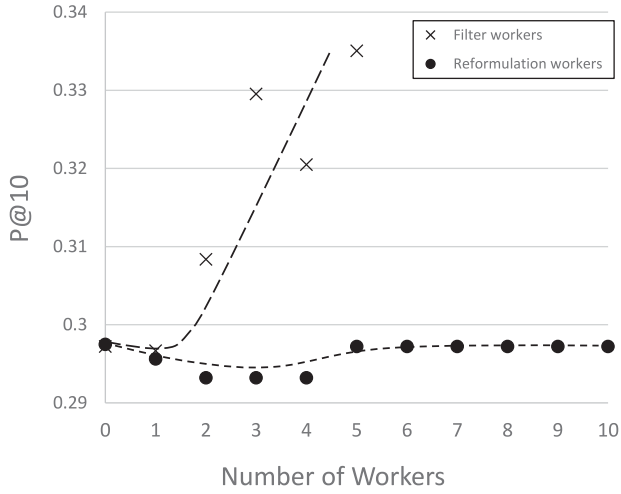
While increasing the number of workers r_n per task gave limited gains in effectiveness (Figure 2(a)), it did increase robustness, visualized by the reduced left-side tail of the gain/loss histogram (Figure 2(b)). The risk-sensitive metric ERR-IA@5 with $\alpha = 10$ in Table II increased significantly from $f_n = 1$ to $f_n = 5$: much more than the risk-agnostic ERR-IA@5. Another interpretation of this result is that increasing the number of workers from 0 to 10 reduced the probability of query expansion failure (performing worse than the baseline) from 14% to 10%.

Figure 2 shows that filtering improved precision roughly in proportion to the number of filter workers: the ability of filtering to mitigate large expansion failures contributed greatly to the overall robustness of the search process.

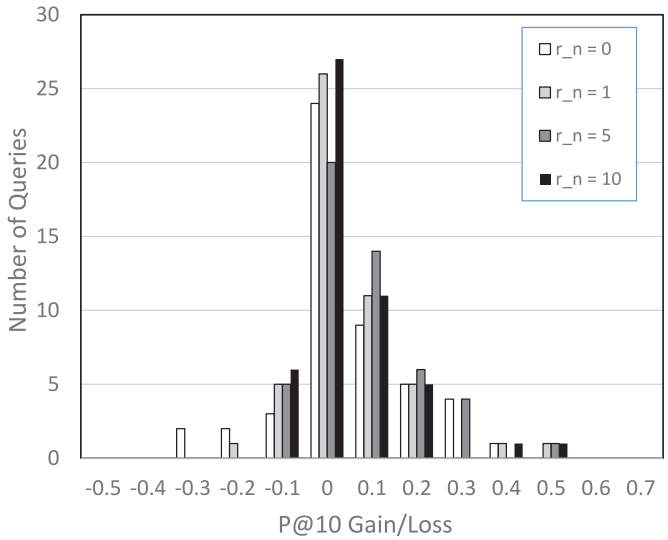
4.4.2. Difficult Web Queries. The results obtained using query reformulation and result filtering for the difficult web queries are summarized in Table III. For comparison, the TREC Web baseline is shown. We also report four combined runs that used either two or five terms to reformulate the query, followed by either no filtering component or filtering using $f_n = 5$ workers.

Overall, using crowdsourcing did not improve the baseline: it significantly hurt P@5 and reduced average ERR@5. This drop appears due almost entirely to the query reformulation component, which (similar to the Web Track queries) reduced effectiveness substantially. The filtering component mitigated this somewhat, but not enough to overcome the losses sustained in query reformulation.

Looking more closely at the poor query reformulation results, one problem was that the difficult queries were hard for even humans to understand. Many of the difficult queries were long and their intents unclear. The crowd workers may have thus been unable to accurately judge the relatedness of a candidate expansion term to the query terms. The difference between the performance of the query reformulation module for the TREC queries and the difficult queries support this view: the crowd workers module was much more effective in the TREC queries, which were much easier to comprehend.



(a) Effect of increasing worker per task. X-axis indicates the number of workers used for the task. Y-axis indicates the P@10 of the resulting ranked list.



(b) Robustness of reformulation results. X-axis indicates the change in P@10 for the queries generated by the reformulation. Y-axis indicates the number of queries which had the specified amount of change.

Fig. 2. Increasing workers per task (top) consistently increased P@10 for the filtering task. While there was little effect on average P@10 for the query reformulation task, the robustness of query expansion improved as workers per task (r_n) increased (bottom). Results shown for TREC 2013 dataset.

Another reason for poor reformulation performance may have been the inability of the search engine query language to make use of the term weights provided by the expansion algorithm and the workers. Without weights, the impact of adding the expansion terms on the overall query was often negative because it is difficult to control the impact the expansion terms has on the overall query. In the case of shorter queries,

Table III. Effectiveness of Difficult Web Query Runs, for Different Combinations of r_k Expansion Terms and Filtered Versus Unfiltered Results

run	ERR@5	P@5
baseline	0.4650	0.3840
$r_k = 2$, unfiltered	0.4298	0.3220†
$r_k = 2$, filtered	0.4437	0.3380†
$r_k = 5$, unfiltered	0.4040	0.3040†
$r_k = 5$, filtered	0.4123	0.3120†

† indicates a statistically significant difference from the baseline using the paired two-tailed t-test with $p < 0.05$.

the expansion terms may overwhelm the original query terms, causing topic drift from the original intent. We believe the optimal weight for expansion terms is fairly small, as was seen in Section 4.4.1, and the fact that the run with $r_k = 2$ performed better supports this view.

4.5. Summary

In this section, we explored the use of crowdsourcing to improve the existing search experience by augmenting traditional search components with crowd intelligence. We saw that crowd-based result ranking increased the accuracy of the final ranked list. While the crowd-augmented query expansion component did not produce significant effective gains, it delivered robust results in the TREC dataset, reducing the number of queries that were harmed by query expansion while maintaining overall accuracy.

This result suggests that crowdsourcing may be useful in sanity-checking intelligent systems such as automated healthcare, where failure incurs a high cost. In addition, the data generated from systems with crowd components could be captured and used as high-quality training data to learn to automate the work that the crowd does. In this way, the crowd can enable search systems to provide robust intelligent functionality prior to having sufficient training data, and then transition to a consistent, reliable automated system. However, due to the monetary and time cost of incorporating crowdsourcing, our results suggest that using crowdsourcing to improve ranking in real time may not be worthwhile in a live search system.

Furthermore, for more difficult queries, crowdsourcing failed to create retrieval effectiveness improvements, and in particular, the query expansion model negatively impacted the accuracy. For many of these queries, even humans had trouble judging related terms, which limited the potential improvements that could be expected from crowdsourcing. In order to produce a more immediate impact on the user's search experience, we must go beyond simple replacements of automated processes and explore methods of incorporating the crowd in the search process that significantly enhance the user's search experience.

5. CROWD-ENHANCED USER EXPERIENCE

We now examine the use of crowdsourcing in ways that improve and change the search experience. Because people are good at understanding complex information needs and synthesizing unstructured information, we focus in this section on a type of query that often requires this type of processing: complex entity queries. Users often express such queries in simplified form: for example, users might search for "drawer pulls" when they really want drawer pulls that have 4" center-to-center spacing in a pewter or nickel finish. Long, attribute-rich queries like "drawer pulls with 4" center-to-center spacing in a pewter or nickel finish" are not well served by current web search engines, which

Table IV. Survey Results for Why Shorter Queries Are Used Instead of More Descriptive Queries (Participants Were Allowed to Select Multiple Answers)

Reason	%
I expect the first query to get better results	38
It takes too long to type out the second query	34
It is too hard to think of what the right attributes are	13
I don't know what attributes I want until after I search	20
The first query just happened to come to mind first	32

typically use more surface-level term-matching strategies, as opposed to the deeper forms of language processing needed to identify and enforce complex constraints.

To understand the nature of authentic attribute-based entity searches, we conducted a survey of 100 interns from a large technology company. Interns were recruited through circulating an email on the general mailing list and are expected to be well versed in the use of search engines. In addition, much of this population had recently moved to a new area that encouraged the types of queries we wanted to investigate.

Participants were asked to recall a recent instance when they wanted to find a specific entity from among a set of candidates, using their search history as necessary. They were then asked to report two versions of a query: one they would type into a search engine (e.g., "hotels in las vegas") and another richer representation that described important attribute requirements (e.g., "four star plus hotels in Las Vegas on the strip which have special discounted rates"). They were also asked why they might issue the shorter query rather than the longer query.

Participants searched for a wide variety of entities, including travel (e.g., a hotel), online shopping (e.g., a space heater), physical stores (e.g., a tire replacement shop), personal finance (e.g., a credit card), and technology (e.g., a media browser). Restaurants emerged as an especially popular topic, at 17% of searches. Generally, users created much shorter queries (3.15 words on average) when asked for the query they would issue to a search engine, as opposed to when they were prompted to give a more descriptive query (7.69 words on average), in line with previous research [Jansen et al. 2000].

When respondents were asked why they preferred to enter the shorter query, 38% replied that they expected the shorter query to find better results than the more descriptive query, as shown in Table IV. Previous work confirms that search engines tend to perform better on short queries [Bendersky and Croft 2009], but the shorter queries that respondents used were often too ambiguous to identify good matches. For example, although one respondent entered "Mt. Baker rental" when searching for a vacation rental, few of the results met the requirements that the rental not allow pets and be close to skiing, and those that did were impossible to identify from the search result page. The short queries respondents entered were underspecified, but the long queries were too complicated for the search engine to algorithmically understand.

To help search engines do a better job with attribute-based entity searches, rather than mimicking existing search engine functionality, we chose to employ crowd workers in a way that fundamentally changes the underlying search experience. In the following sections, we detail the components we studied and describe the experimental setup. We then delve into a discussion of the experimental results, analyzing the impact of experimental variables and characteristics of queries on search effectiveness. The impact on user experience is explored separately through a user survey.

5.1. Query Understanding: Identifying and Matching Query Attributes

The query understanding component seeks to correctly segment, understand, and use the components of the query, in three processing stages. In the first stage, crowd workers

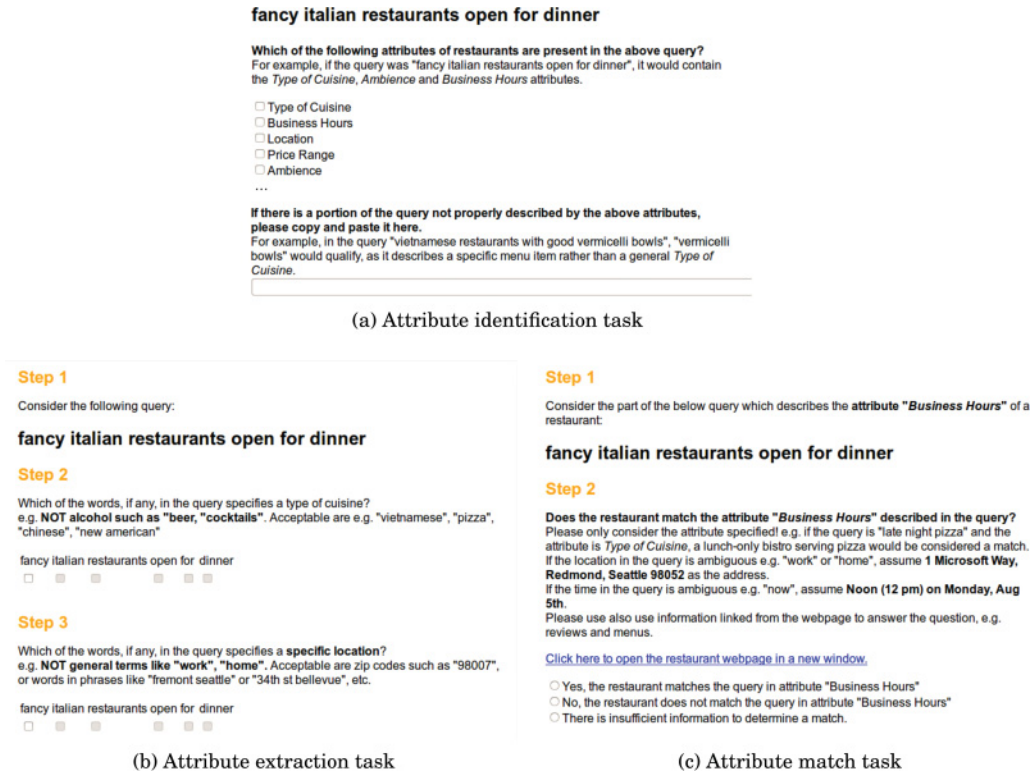


Fig. 3. Interface design of the crowd tasks.

perform *attribute identification*, identifying which attributes are present in the user's query (Figure 3(a)). For example, in the query "Italian restaurant in Seattle open for dinner on Sunday," the workers would identify the attributes: type of cuisine ("Italian"), location ("Seattle"), and business hours ("open for dinner on Sunday"). The crowd workers can identify these attributes from a given domain-specific list, name the attributes themselves, or do a combination of both, to ensure high-quality answers for common attributes and to cover attributes not addressed by the list. Domain-specific lists can be created from sources such as Yelp Features, which lists attributes such as business hours and locations for restaurants. Note that to maintain the simplicity of this task, workers are merely asked about the *presence* of attributes in the first stage: they do not need to provide the mapping of each attribute to its corresponding query terms.

Once the presence of attributes is identified, the second *attribute extraction* stage (Figure 3(b)) asks crowd workers to extract, for a given attribute, the specific term or terms in the user's original query that match the attribute. This is done for each of the attributes identified by workers in the first stage. In the previous example, "Italian" would be extracted for the type of cuisine attribute. The number of workers for the attribute identification and attribute extraction tasks can vary: our experiments used 10 workers for each task instance.

Finally, in the third *candidate retrieval* stage, a "relaxed" form of the original query is created that removes all but the most salient attributes for the particular domain and is then submitted to a search engine. The purpose of this stage is to identify a pool of candidate results that is likely to contain relevant documents satisfying further constraints. This step is domain specific: for example, in the restaurant domain, salient

attributes might be the restaurant's location and type of cuisine, so that an original query "Italian restaurant in Seattle open for dinner on Sunday" would be relaxed to the query "Italian restaurants in Seattle." The original query, the attribute annotations, and the candidate result list retrieved by the relaxed query are then passed to the result understanding component for further processing, described next.

5.2. Result Understanding: Creating Table-Based Summaries

One of the main advantages of the deeper understanding provided through crowdsourcing is the ability to use the detailed information to create new user experiences.

In the result understanding stage, relevant results from the candidate list are identified and summarized into a table-like view. Two example table views, for a restaurant query and a shopping query, are shown in Figure 4 in Section 5.3.4.

To create these views, workers perform an *attribute match* task (Figure 3(c)) in which a candidate document is checked to see if it satisfies a particular query attribute. This is done for all combinations of candidate documents and query attributes. While it is hard to do this algorithmically for an arbitrary attribute (such as "has valet parking" or "serves kale"), it is something people can do easily. The workers' results are aggregated, producing a table view in which each row lists a result, and the columns indicate whether the attribute requirements are satisfied. If there is disagreement among workers about the attribute, the attribute is marked as "unsure." Results with more matching attributes are ranked higher, and ties are broken so results with more positive votes overall from the workers are ranked higher. The number of workers per task is a parameter: we explore varying this number in our experiments between one and five workers per result-attribute pair.

As Figure 4 suggests, our approach can generalize to a wide variety of entity search tasks with some additional effort. One stage, the query relaxation step for candidate retrieval, has a domain-dependent implementation. However, it does not require a domain-dependent corpus to be created, which would be prohibitively time-consuming and expensive. Instead, given an index that returns entities, such as Freebase, our crowd-driven attribute-based search process could be used to improve search for entities in any domain. We leave the application and evaluation of this approach in different domains to future work.

5.3. Experimental Results: Crowd-Enhanced User Experience

In this section, we describe the queries and dataset used and present the results of our experimentation. We evaluate the crowd-enhanced search experience in two stages. First, in Section 5.3.3, we examine retrieval effectiveness on a set of authentic restaurant queries, using traditional information retrieval metrics. We also examine how effectiveness varies with the number of crowd workers and across subgroups of queries with different location attributes. Second, in Section 5.3.4, we describe a user study that evaluates the effect of these components on the user experience. These two elements are studied separately so that preferences for the user experience are not biased by the effectiveness of results. As explained in Section 4.3, an exploration of time and monetary tradeoffs is omitted here and is a topic for future work.

5.3.1. Queries. Although users sometimes write long queries, most people intentionally keep their queries short, even for complex needs, as our survey in Section 5 showed. For this reason, we evaluated our system using search queries collected outside of a traditional search framework. Interns at a large technology company were recruited to participate in an online survey where they were asked to create a query which described a type of restaurant they would like to visit, using natural language and multiple attributes (such as location, business hours, and type of cuisine). The queries

Table V. Statistics of the Restaurant Query Set

# of Queries	Avg Terms in Query	Shortest	Longest
70	9.57	2 terms	29 terms

gathered from this survey were manually filtered to remove spam and errors, resulting in 70 queries. These queries had a wide range of query lengths (Table V). Examples of queries gathered include "thai, greek, late night, near work" and "moderately priced sushi restaurant open later than average." The full list of queries may be found online.[6]

5.3.2. Dataset. To evaluate search effectiveness for our restaurant queries, we used Yelp[7] as the source for our candidate documents. We selected Yelp for its focused coverage of restaurant entities, but as mentioned previously, our approach does not actually require a domain-specific corpus.

In the attribute identification task (Figure 3(a)), workers were provided with a default attribute list constructed based on Yelp Features. Workers could also add new attributes that they felt were not covered by the default attribute list. For the attribute extraction task, workers identified the query terms corresponding to the attributes for location and type of cuisine (Figure 3(b)). For more robust results, our experiments used 10 workers per each task instance.

The candidate retrieval stage used these terms to construct the relaxed query, which was submitted to the Yelp API to obtain the top 10 results as the set of candidates. The cuisine term in the relaxed query defaulted to "restaurant" if no cuisine type was identified, and the location term defaulted to the address of the technology company when no absolute location was specified, or if a relative location (such as "near me") was specified. Note that the crowd workers merely extracted the parts of the query that referred to a location: they did not attempt to determine the real, physical location associated with the text. The physical location was determined through the default location or through the Yelp API's internal location matching mechanism. Finally, as described in Section 5.2, a crowd task was created to identify attribute matches between the query and a candidate result (Figure 3(c)).

For comparison, we defined a baseline retrieval method that submitted the user's unprocessed query to a major search engine, limiting the results to Yelp business pages. (This use of an external search engine was necessary because the character limit on queries with Yelp's API was shorter than many of our queries.) Despite the natural language descriptions in many queries, this baseline often found relevant matches within the descriptions of user reviews associated with the business page and returned surprisingly accurate results.

5.3.3. Analysis of Effectiveness. To evaluate the effectiveness of the results, we use standard retrieval metrics such as expected reciprocal rank (ERR), precision, and mean average precision (MAP). Binary (relevant or nonrelevant) ground-truth relevance judgments were assigned to all results by one of the authors. As a reliability check, a second author independently judged a sample of approximately 10% of the query-URL pairs: the resulting Cohen's kappa score was 0.77, which indicated substantial agreement between the raters.

Table VI compares the effectiveness of the attribute-based crowdsourced approach with the baseline automated approach in terms of P@5, ERR@5, and MAP. (Recall that the baseline method returns Yelp business pages returned by a commercial search engine in response to the user's original query.) The crowdsourced method outperformed

[6]www.cs.cmu.edu/∼yubink/restaurant_queries.txt.
[7]www.yelp.com.

Table VI. Overview of Results for Entity Attribute Queries

Run	ERR@5	P@5	MAP
baseline	0.4108	0.3257	0.2877
initial	0.4478	0.3829	0.3369
1 worker	0.4148	0.3618	0.3267
2 workers	0.4473	0.3765	0.3407
3 workers	0.4743	0.4235	0.3448
4 workers	0.4649	0.4206	0.3465
final	0.4685	0.4206†	0.3468

The initial run is results retrieved from Yelp without any reranking based on attribute matches. The final run is reranked based on the votes of up to 5 crowd workers per result. † indicates statistical significance against the baseline using the paired two-tailed t-test with $p < 0.1$.

Table VII. Accuracy of Baseline and Final Crowdsourced Results for Three Types of Queries

Query Type	#	Run	ERR@5	P@5	MAP
specific location	36	baseline	0.4838	0.3611	0.3396
		final	0.4530	0.4000	0.3480
relative location	15	baseline	0.0000	0.0000	0.0067
		final	0.4008†	0.4133†	0.4184†
no location	19	baseline	0.5968	0.5158	0.4113*
		final	0.5019	0.4211	0.2516

† indicates statistical significance ($p < 0.005$) using the paired two-tailed t-test. * indicates statistical significance with $p < 0.1$.

the baseline method on average across all measures: ERR@5 increased by 14%, P@5 by 29%, and MAP by 21%. The difference in P@5 was statistically significant at the 10% level using a two-tailed paired t-test. Differences in ERR@5 and MAP were not statistically significant with this query sample, which exhibited high variance in baseline effectiveness: some queries had many relevant matches, while others had none.

Impact of Additional Workers. Table VI also shows the increase in effectiveness that results from increasing the number of workers used to match a candidate result to query attributes. The "initial" run denotes the set of search results retrieved from Yelp using the reduced query containing only the cuisine type and location. Subsequent rows show the result of applying crowd-based reranking based on the attribute matches produced by n workers per result. In general, evaluation measures show improved effectiveness as more workers are added, in accord with the trends seen with result-filtering workers in Section 4.4. The "final" run, using five worker labels per result, is significantly higher in effectiveness than the initial retrieval. The drop in accuracy with one worker could be considered an artifact of chance variations in individual worker quality for this particular task and query set.

Impact of Location. During analysis, we noticed a distinct trend in our results when the queries were separated based on the characteristics of the location attribute. We looked at queries with location specified three different ways: (1) with specific locations (e.g., "redmond," "98074"), (2) with relative locations (e.g., "nearby," "10 mins walk"), and (3) with no specified location. Table VII compares the baseline method and the crowdsourced method for each of the three types.

The baseline performance was better for queries with no locations. This is due to the limitations of the Yelp API used for the crowdsourced run: the API requires that a location is always specified in the search, thus limiting the pool of restaurants. The

Table VIII. The Effect of Crowd-Added Attributes on Accuracy of the Reranking Process

Run	ERR@5	P@5	MAP
without crowd-added attributes	0.3867	0.3000	0.2568
with crowd-added attributes	0.4145	0.3182	0.2484

Evaluation metrics were calculated only over the 22 queries where crowd attributes were added.

baseline method had no such constraints and was able to search restaurants from all over the country, drawing from a much larger pool of candidates. For example, for the query "sandwich place that has a caprese sandwich," the Yelp API returned zero possible matches, while the baseline identified a number of potential candidates.

For queries that include a specific location, the differences in retrieval effectiveness were not statistically significant. The baseline queries found matches for its attributes against the text of the user-written reviews and often produced good results.

However, in queries with relative locations, the crowdsourced method performed far better than the baseline (0.4 vs. 0 in ERR@5). This is due to the fact that through the crowdsourced annotations, the system is able to leverage the knowledge that these queries mention a location attribute, but no specific location. Based on this, the system can limit the results of the query to the geographic location of the query issuer. This analysis suggests that one of the big strengths of crowdsourced attribute analysis is the ability to accurately identify when default values should be applied.

Impact of Crowdsourced Attributes. During the query attribute annotation step (Section 5.1), new attributes were created by the crowd when the predefined list of attributes did not include attributes mentioned in the user's query. New crowd-created attributes added to the query set included "nut allergy-safe," "has chicken fettuccini alfredo," and "has great service." There were a total of 22 queries where these extra attributes were added.

Table VIII summarizes the effect that the extra attributes had on the retrieval effectiveness of the reranked results for this query segment. There were slight improvements on ERR@5 and P@5, but these were not statistically significant. However, the extra attributes turned out to improve the user experience in other ways, as described in Section 5.3.4. Interestingly, the effectiveness measures are lower than average for these queries (compare Table VIII with Table VI). Many long and difficult queries in the query set were assigned extra attribute annotations: their average query length was 11.6 terms, compared to 9.6 terms in the overall query set. Seen in context with the aforementioned results, it suggests that longer, more difficult queries could benefit from crowdsourced processing, perhaps more so than easier queries.

5.3.4. Analysis of User Experience. To evaluate how people responded to a tabulated interface, we ran a user study comparing a traditional list of results with a tabular, attribute-annotated version of the same set of results. Users were shown the output of the crowdsourced search for two different randomly selected queries. For the first query, results were presented using a traditional list-like result page (Figure 5). For the second query, results were displayed in a tabular format (Figure 4(a)). These result pages were presented in sequence: from a start page, the user would first click to view one result page, return to the start page, then click to view the second result page. To control for potential learning or primacy effects, half of the participants were shown the list-like result page first, and the other half were shown the table-like result page first.

The list and tabular versions of results for any given query used the same underlying set of result entities (e.g., restaurants), so that only the presentation method was being

centric restaurant in manhattan, close to the empire state building. Argentine or Mexican food. Quiet place. With reservation.

	Takes Reservations	Type of Cuisine	Location
Azul Bistro - New York, NY	✓	✓	✓
El Gauchito - Elmhurst, NY	✓	✓	✓
El Almacen - Brooklyn, NY	✓	✓	✓
Pampas Argentinas - Forest Hills, NY	✓	✓	✓
Sosa Borella - New York, NY	✓	✓	✓
Libertador - New York, NY	✓	✓	✓
Mexico Lindo Restaurant - New York, NY	✓	✓	✓
El Mariachi Restaurant - Astoria, NY	?	✓	✓
Nuchas - New York, NY	✗	✓	✓
Empanadas Bar NYC - New York, NY	✗	✓	✓

(a) Example restaurant query

drawer pulls with 4" center-to-center spacing in a pewter or nickel finish

	Spacing	Finish
Giagni CP-4-SN Contemporary Cup Pull in Satin Nickel	✓	✓
Liberty Satin Nickel 4 in. Notched Pull	✓	✓
Top Knobs M1190 Pull 4" Center-to-Center Pewter Antique	✓	✓
stainless steel cabinet handle Brush Nickel Kitchen Cabinet Pull Handle knob 4"	✓	✓
Century Hardware 22197 Regal Zinc Die Cast 4 Arch Pull	✓	✗
JVJ Hardware 66346 Pewter 3 3/4-inch Leafed Edge Pull	✗	✓

(b) Example product query

Fig. 4. Table-like view of results generated by the crowd.

cheap restaurant somewhere around my hotel or work, that serves very good baked pork chop, lor bak or pizza and is open for at least two hours

1. Steelhead Diner - Seattle, WA

   718 reviews

   We had a fabulous lunch here on our last day in Seattle! Service was top-notch and efficient. We asked for patio seating and we had a table within 5...

2. Toulouse Petit - Seattle, WA

   1364 reviews

   let me preface this by saying i'm a very picky bellini drinker. i usually prefer very dry high quality champagne and just the right balance of fresh...

3. Palace Kitchen - Seattle, WA

   605 reviews

   My favorite restaurant in Seattle so far. The wait staff is none short of amazing and the food is delicious. My go to entrees are the burger and the...

Fig. 5. List-like view of results shown to user study participants.

Table IX. Summary of Study Participants' Preferences for the Two Presented Interfaces

Preference	# of People
List: Strong Preference	2
List: Weak Preference	1
Neutral	2
Table: Weak Preference	3
Table: Strong Preference	8

modified, not the set of result entities. In addition, the list interface mimicked the result list from the major search engine used to collect the results. This means that the list contained information not included in the tabular format, including a textual snippet and the Yelp star rating for the restaurant.

For each query, the participants were asked to browse the results and find a restaurant that was a good match for the given query. Afterward, a survey asked for their preferences and reasons, and prompted participants to suggest additional useful attributes. The participants in the study were 16 university students: 13 male and three female, ranging in age from from 23 to 40 years old.

The results of the user study are summarized in Table IX. Most participants preferred the rich search result understanding interface provided by crowd workers over the traditional list interface. Eleven participants (69%) said they "strongly" or "weakly" preferred the table, while only three (19%) said the same for the list. When asked to elaborate why they preferred the table interface, respondents reported that the summary view was useful because they could eliminate certain restaurants without having to click through to the result page. (The difference in actual times was not statistically significant due to variance in queries.) On the other hand, people chose the list interface because of familiarity (e.g., "Google-like," "familiar scan pattern").

Although the table view of the result list displayed the attributes requested in the query, participants sometimes requested other attributes. The most common suggestions were for the star ratings and reviews for the restaurants, mentioned by 37.5% of participants. Interestingly, participants favored the table view over the list view even though the list view displayed rating and review information. The participants' feedback seems to suggest that in most restaurant queries, there is an implicit attribute of "quality." Such implicit attributes are obviously domain dependent: exploring implicit attributes of other domains would be a valuable extension to this work.

Impact of Crowdsourced Attributes. In a second small-scale study, we studied the effects of crowd-created attributes on user experience. The 22 queries that had crowd-created attributes were given to a group of seven university-aged users. After interacting with the table to find a restaurant matching the query, the participants were asked to identify the single most helpful attribute. We found that for 11 (50%) of the provided queries, participants stated that one of the extra attributes was the most useful attribute. Given that each query had an average of 3.43 attributes each, this indicates that extra attributes were disproportionately useful to the users.

5.4. Summary

The crowdsourcing modules described in this section aimed to improve search by deviating from standard search components to deliver a novel user experience. The resulting new system led to gains in both search effectiveness and user experience. We found that the greatest improvements in effectiveness were in queries with relative location: unlike the automated baseline system, the crowdsourced system was able to correctly identify these queries and assign the right default values to the location

attribute. We also saw that most users preferred the table-like summary generated by the crowdsourcing modules over a list view, and identified useful domain-specific implicit attributes. Finally, although crowd-generated attributes did not materially contribute to gains in search effectiveness, users considered them very helpful in their task.

Typically, systems incorporating crowdsourcing are not able to scale to a free-to-use, live search system as the cost would be prohibitive. However, there are other areas in which a crowd-based system may be successful. One example is a pay-for-search model such as Jisiklog [Lee et al. 2013]. Also, this system can be implemented in a "friendsourcing" environment where questions are asked of friends with social reciprocity as payment [Jeong et al. 2013]. Additionally, some of the crowdsourced processes explored in this paper could also be used to selfsource the search experience by helping users themselves structure their search processes [Teevan et al. 2014].

Furthermore, this work is a demonstration of the value of crowdsourcing in rapidly prototyping and evaluating new search interfaces and experiences. Implementing a similar system using automatic components would have been difficult, as it would require highly accurate analysis of complex or loosely structured natural language queries and, more importantly, a clean, structured entity database with many defined fields. Crowdsourcing essentially created this database on the fly with the ability to add new fields as requested by the user. This in turn allowed us to evaluate a new search interface without expensive data collection. In addition, the data generated from the crowdsourcing experiments may be used to train an automatic system that would run at scale, which is an interesting direction for future work.

6. CONCLUSIONS

In this article, we explored the use of crowdsourcing to incorporate human computation into the web search pipeline in order to overcome existing limitations of automated components and make web search more intelligent. We used a framework that breaks the search process down into distinct stages that can be supported with either algorithmic or crowdsourced components. We studied instantiations of this framework that used crowdsourced components for query and result understanding to improve search result ranking, and to change the existing search experiences.

We found that crowdsourcing could improve search result ranking, and that crowd-augmented query reformulation and result list filtering resulted in a more reliable search experience than automated approaches. This is probably because humans are less likely than algorithms to mistake irrelevant but correlated content as relevant to a query. Our findings suggest that humans may make good replacements for automated processes in systems where errors are costly or where sufficient training data does not yet exist. However, automated approaches to the information retrieval modules that we explored are already well understood, and the overall gains we observed appear insufficient to justify the additional time and expense required to incorporate the crowd.

In contrast to the ranking-centric approaches we explored, we found that using the crowd to explore a significantly different and intelligent search experience yielded noticeable benefit. We implemented crowd-based attribute annotation and table summarization, and observed that these components led to improvements in search effectiveness and user experience. We identified queries where default values should be applied, discovered important default attributes, and showed that crowd-generated attributes were valued by users. Additionally, the table view generated by the result composition module was preferred by most participants over the traditional search result list view.

By incorporating the crowd into web search, we were able to break away from focusing on limitations of algorithmic accuracy and concentrate directly on how a smart search system with a significantly different approach might help users. There is an opportunity for designers of complex systems like web search to use the crowd to quickly build and prototype riskier, more creative solutions than they might otherwise, and receive immediate feedback on its performance. The data collected from successful experiments can then be used to guide the development of reliable automated components.

Our results confirm that taking more time and including human computation in the search process can lead to improvements in search effectiveness, robustness, and user experience. Looking ahead, our findings suggest that significant changes to existing paradigms may be needed for crowdsourcing to fulfill its potential in search systems.

REFERENCES

Omar Alonso, Daniel E. Rose, and Benjamin Stewart. 2008. Crowdsourcing for relevance evaluation. In *ACM SIGIR Forum*, Vol. 42. 9.

Michael Bendersky and W. Bruce Croft. 2008. Discovering key concepts in verbose queries. In *Proc. SIGIR*. ACM, New York, NY, 491–498.

Michael Bendersky and W. Bruce Croft. 2009. Analysis of long queries in a large scale search log. In *Proc. WSCD*. ACM, New York, NY, 8–14.

Michael S. Bernstein, Greg Little, Robert C. Miller, Björn Hartmann, Mark S. Ackerman, David R. Karger, David Crowell, and Katrina Panovich. 2010a. Soylent: A word processor with a crowd inside. In *Proc. UIST*. ACM, New York, NY, 313–322.

Michael S. Bernstein, Jaime Teevan, Susan Dumais, Dan Liebling, and Eric Horvitz. 2010b. Direct answers for search queries in the long tail. In *Proc. CHI*. 237–246.

Alessandro Bozzon, Marco Brambilla, and Stefano Ceri. 2012. Answering search queries with crowdsearcher. In *Proc. WWW*. ACM, New York, NY, 1009–1018.

Claudio Carpineto, Stanislaw Osiński, Giovanni Romano, and Dawid Weiss. 2009. A survey of web clustering engines. *ACM Comp. Surv.* 41, 3, Article 17 (2009), 38 pages.

Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proc. CIKM*. 621–630.

Xi Chen, Paul N. Bennett, Kevyn Collins-Thompson, and Eric Horvitz. 2013. Pairwise ranking aggregation in a crowdsourced setting. In *Proc. WSDM*. 193–202.

Kevyn Collins-Thompson. 2009. Reducing the risk of query expansion via robust constrained optimization. In *Proc. CIKM*. 837–846.

Kevyn Collins-Thompson and Jamie Callan. 2005. Query expansion using random walk models. In *Proc. CIKM*. 704–711.

Gordon V. Cormack, Mark D. Smucker, and Charles L. A. Clarke. 2011. Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retrieval* 14, 5 (2011), 441–465.

Daniel Crabtree. 2007. Exploiting underrepresented query aspects for automatic query expansion categories and subject descriptors. In *KDD*. 191–200.

Gianluca Demartini, Beth Trushkowsky, Tim Kraska, and Michael J. Franklin. 2013. CrowdQ: Crowdsourced query understanding. In *Proc. CIDR*.

Susan T. Dumais. 2013. Task-based search: A search engine perspective. Talk at NSF Task-Based Information Search Systems Workshop. (March 14–15, 2013). Retrieved from http://bit.ly/15rK5tD.

Michael J. Franklin, Donald Kossmann, Tim Kraska, Sukriti Ramesh, and Reynold Xin. 2011. CrowdDB: Answering queries with crowdsourcing. In *Proc. SIGMOD*. ACM, New York, NY, 61–72.

Michael J. Franklin, Beth Trushkowsky, Purnamrita Sarkar, and Tim Kraska. 2013. Crowdsourced enumeration queries. In *Proc. ICDE*. IEEE Computer Society, Washington, DC, 673–684.

Brent Hecht, Jaime Teevan, Meredith Ringel Morris, and Daniel J Liebling. 2012. SearchBuddies: Bringing search engines into the conversation. In *Proc. ICWSM*, Vol. 12. 138–145.

Gary Hsieh and Scott Counts. 2009. Mimir: A market-based real-time question and answer service. In *Proc. CHI*. 769–778.

Panagiotis G. Ipeirotis, Foster Provost, and Jing Wang. 2010. Quality management on amazon mechanical turk. In *Proc. HCOMP*. ACM, New York, NY, 64–67.

Bernard J. Jansen, Amanda Spink, and Tefko Saracevic. 2000. Real life, real users, and real needs: A study and analysis of user queries on the web. *IP&M* 36, 2 (2000), 207–227.

Jin-Woo Jeong, Meredith R. Morris, Jaime Teevan, and Daniel Liebling. 2013. A crowd-powered socially embedded search engine. In *Proc. ICWSM*.

In-Ho Kang and Gil Chang Kim. 2003. Query type classification for web document retrieval. In *Proc. SIGIR*. ACM, New York, NY, 64–71.

Gabriella Kazai, Jaap Kamps, Marijn Koolen, and Natasa Milic-Frayling. 2011. Crowdsourcing for book search evaluation: Impact of hit design on comparative system ranking. In *Proc. SIGIR*. ACM, New York, NY, 205–214.

Giridhar Kumaran and Vitor R. Carvalho. 2009. Reducing long queries using query quality predictors. In *Proc. SIGIR*. ACM, New York, NY, 564–571.

Walter S. Lasecki and Jeffrey P. Bigham. 2014. Interactive crowds: Real-time crowdsourcing and crowd agents. In *Handbook of Human Computation*.

Victor Lavrenko and W. Bruce Croft. 2001. Relevance based language models. In *Proc. SIGIR*. ACM, New York, NY, 120–127.

Edith Law and H Zhang. 2011. Towards large-scale collaborative planning: Answering high-level search queries using human computation. In *Proc. AAAI*.

Uichin Lee, Jihyoung Kim, Eunhee Yi, Juyup Sung, and Mario Gerla. 2013. Analyzing crowd workers in mobile pay-for-answer Q&A. In *Proc. CHI*. 533–542.

Adam Marcus, Eugene Wu, Samuel Madden, and Robert C. Miller. 2011. Crowdsourced databases: Query processing with people. In *Proc. CIDR*. CIDR, 211–214.

Winter Mason and Duncan J. Watts. 2009. Financial incentives and the "performance of crowds". In *Proc. HCOMP*. ACM, New York, NY, 77–85.

Aditya G. Parameswaran, Ming Han Teh, Hector Garcia-Molina, and Jennifer Widom. 2013. DataSift: An expressive and accurate crowd-powered search toolkit. In *Proc. HCOMP*.

Matthew Richardson and Ryen W. White. 2011. Supporting synchronous social q&a throughout the question lifecycle. In *Proc. WWW*. 755–764.

Nikos Sarkas, Stelios Paparizos, and Panayiotis Tsaparas. 2010. Structured annotations of web queries. In *Proc. SIGMOD*. ACM, New York, NY, 771–782.

Eric Schurman and Jake Brutlag. 2009. Performance related changes and their user impact. Velocity. (2009). http://oreil.ly/fTmYwz.

Jaime Teevan, Kevyn Collins-Thompson, Ryen White, Susan Dumais, and Yubin Kim. 2013. Slow search or: How search engines can learn to stop hurrying and take their time. In *Proc. HCIR*. ACM, New York, NY.

Jaime Teevan, Kevyn Collins-Thompson, Ryen W. White, and Susan Dumais. 2014. Slow search. *Commun. ACM* 57, 8 (2014), 36–38.

Jaime Teevan, Daniel J. Liebling, and Walter S. Lasecki. 2014. Selfsourcing personal tasks. In *Proc. CHI EA*. 2527–2532.

Tingxin Yan, Vikas Kumar, and Deepak Ganesan. 2010. CrowdSearch: Exploiting crowds for accurate real-time image search on mobile phones. In *Proc. MobiSys*. ACM, New York, NY, 77–90.

Haoqi Zhang, Edith Law, Rob Miller, Krzysztof Gajos, David Parkes, and Eric Horvitz. 2012. Human computation tasks with global constraints. In *Proc. CHI*. ACM, New York, NY, 217–226.

Received February 2015; revised August 2015; accepted November 2015