CRQA: Crowd-powered Real-time Automatic Question Answering System

Denis Savenkov
Emory University
dsavenk@emory.edu

Eugene Agichtein
Emory University
eugene.agichtein@emory.edu

Abstract

Modern search engines have made dramatic progress in answering questions about facts, such as those that might be retrieved or directly inferred from a knowledge base. However, many other real user questions are more complex, such as requests for opinions, explanations, instructions or advice for a particular situation, and are still largely beyond the competence of the computer systems. As conversational agents become more popular, QA systems are increasingly expected to handle such complex questions, and to do so in (nearly) real-time, as the searcher is unlikely to wait longer than a minute or two for an answer. One way to overcome some of the challenges in complex question answering is crowdsourcing. We explore two ways crowdsourcing can assist a question answering system that operates in (near) real time: by providing answer *validation*, which could be used to filter or re-rank the candidate answers, and by *creating* the answer candidates directly. In this paper we present CRQA, a crowd-powered, near real-time automatic question answering system for complex informational tasks, that incorporates a crowdsourcing module for augmenting and validating the candidate answers. The crowd input, obtained in real-time, is integrated into CRQA via a learning-to-rank model, to select the final system answer. Our large-scale experiments, performed on a live stream of real users questions, show that even within a one minute time limit, CRQA can produce answers of high quality. The returned answers are judged to be significantly better compared to the automatic system alone, and even are often preferred to answers posted days later in the original community question answering site. Our findings can be useful for developing hybrid human-computer systems for automatic question answering and conversational agents.

Introduction

It has long been a dream to communicate with a computer as one might with another human being using natural language speech and text. We are now coming closer to this dream, as natural language interfaces become increasingly popular. Our phones are already reasonably good at recognizing speech, and personal assistants, such as Apple Siri, Google Now, Microsoft Cortana, Amazon Alexa, *etc.*, help us with everyday tasks and answer some of our questions. "Chat bots", or natural language agents, are being increasingly adapted for business and entertainment, and a number of startups developing this kind of technology have emerged[1].

Question answering is one of the major components of such personal assistants. Existing techniques already allow users to get direct answers to their factoid questions. However, there is still a large number of more complex questions, such as advice or accepted general opinions, for which users have to dig into the "10 blue links" and extract or synthesize answers from information buried within the retrieved documents. To cater to these informational needs, community question answering (CQA) sites emerged, such as Yahoo! Answers and Stack Exchange. These sites provide a popular way to connect information seekers with answerers. Unfortunately, it can take minutes or hours, and sometimes days, for the community to respond, and some questions are left unanswered altogether.

To facilitate research on this challenge, a series of TREC LiveQA evaluation campaigns[2] was introduced in 2015, where automatic systems attempt to answer real user questions within a 1 minute period. This task was successful, with the winning system able to automatically return a reasonable answer to more than half of the submitted questions, as assessed for TREC by the trained judges from NIST. Nevertheless, many questions were not answered well by any of the participating systems (Agichtein et al. 2015).

In this work we explore two ways *crowdsourcing* can be used to help an automatic system answer complex user questions. The main challenge is how to adapt existing "batch-mode" crowdsourcing platforms such as Amazon Mechanical Turk to real-time settings, *e.g.,* to produce an answer within a minute. More specifically, our research questions can be stated as: **RQ1**. Can crowdsourcing be used to improve the performance of a near real-time automatic question answering system? **RQ2**. What kind of contributions from crowd workers can help improve automatic question answering and what is the relative impact of different types of feedback to the overall question answering performance? **RQ3**. What are the trade-offs in performance, cost, and scalability of using crowdsourcing for real-time question answering?

[1] http://time.com/4194063/chatbots-facebook-messenger-kik-wechat/

[2] http://www.trec-liveqa.org

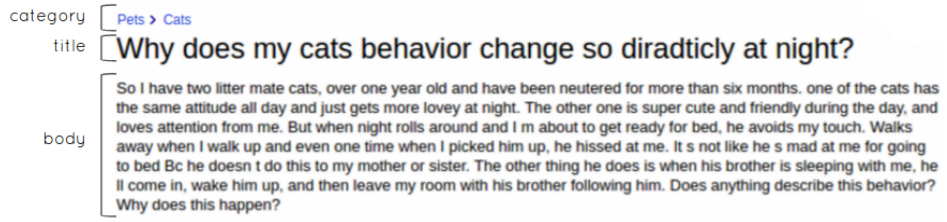


Figure 1: Example of the question from Yahoo! Answers community question answering platform

To explore these research questions, we introduce our CRQA system, which stands for Crowd-powered Real-time Question Answering. CRQA integrates a crowdsourcing module into an automatic question answering system within an overall learning-to-rank framework for selecting answers to complex questions. We report extensive experiments of stress-testing the CRQA system, by participating in the TREC LiveQA 2016 evaluation challenge. Specifically, the contributions of this paper are threefold:

1. CRQA, a novel hybrid near real-time question answering system, that uses crowd workers to rate and augment automatically generated candidate answers, developed to empirically explore the research questions above.
2. Large scale experiments using CRQA to answer real user questions in a live TREC LiveQA 2016 challenge setting, addressing **RQ1**.
3. Extensive analysis of the system performance, focusing on the contributions of crowd input to the performance of the overall system, to explore **RQ2** and **RQ3**.

The results and findings of this work can be useful to guide future research on hybrid human-computer question answering, and automatic intelligent assistant systems.

System design

Before diving into the architecture of our Crowd-powered Real-time Question Answering (CRQA) system, we will describe the setup of the TREC LiveQA shared task, which affected some of the system design choices. In 2015 and 2016 versions of the task participants of the challenge developed a live question answering system, that responded to user questions, which were sampled from the live stream of Yahoo! Answers community question answering website. Each input question consisted of a short question title, body and category (Figure 1). A QA system had to provide an answer of 1000 characters or less within a 1 minute period using any available data source. A reader can refer to (Agichtein et al. 2015) for more details on TREC LiveQA 2015 results and analysis.

Our CRQA system represents a hybrid system, which includes an automatic question answering and crowdsourcing modules. The high level architecture is presented in Figure 2. The automatic part of the CRQA system follows an Information Retrieval (IR) approach to question answering, and generates a set of candidate answer passages from multiple data sources. After candidates are generated, they are ranked by a trained model, and, in the fully automatic mode, the top candidate could be returned as the answer. The crowdsourcing module is designed to overcome two of the most common problems of the automatic QA approaches: lack of good candidate answers and ranking errors (Moldovan et al. 2003; Savenkov 2015). More particularly, CRQA asks crowd workers to provide answers to questions if they can, and additionally rate the quality of candidate answers. These contributions are then used to re-rank the candidates and return the top scoring answer. The next two sections describe the architectures of the automatic and crowdsourcing modules of our system.

Automatic question answering module

When CRQA receives a question, it generates a set of search queries to retrieve a set of relevant documents and extract candidate answer passages. Search queries are generated using the following strategies:

- Question title, which most often captures the gist of the question
- Two longest question sentences (detected by the presence of the question word at the beginning or question mark at the end of a sentence) from the title and body of the question. In some cases the real user question is hidden inside the body, while the title just provides the overall topic of the question.
- Concatenation of the question word, verbs and top-5 terms from the question title by inverse document frequency[3]. This strategy targets over-specific questions (*e.g.,* Figure 1), which often retrieve few if any search results.

To retrieve a set of potentially relevant documents and extract candidate answer passages, CRQA relies on multiple different generic and CQA document collections. Previous research has shown that many of the user information needs are repeated, and reusing answers to previously posted similar questions is an effective strategy for answering new questions (Carmel, Shtalhaim, and Soffer 2000; Shtok et al. 2012). Therefore, CRQA uses multiple different CQA data sources, namely Yahoo! Answers, Answers.com and WikiHow.com, which potentially contain a diverse set of questions. To retrieve similar questions we use the built-in search interfaces of the corresponding websites. CRQA

[3]IDF of terms are estimated using Google N-gram corpus: https://catalog.ldc.upenn.edu/LDC2006T13

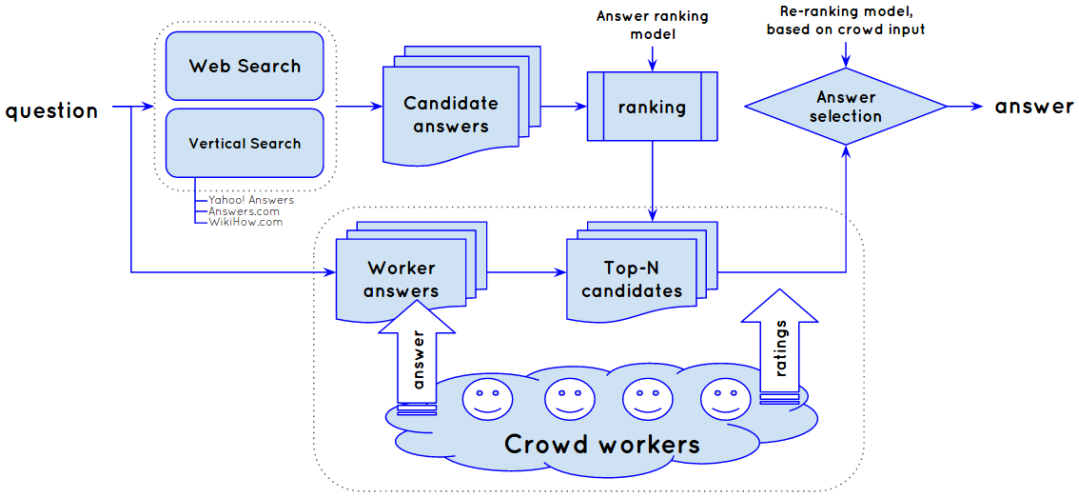


Figure 2: The architecture of our Crowd-powered Real-time Question Answering system, that uses crowdsourcing to augment a list of automatically extracted candidate answers and to rate their quality

extracts top-10 similar questions and the corresponding answers, posted by the community, and adds them to the pool of candidate answers. However, quite often it is hard to find a similar question in an archive, and many of the information needs are unique. Therefore, we integrate web search[4], which our system queries to retrieve candidate answers from regular web documents. We retrieve top-10 relevant web documents and extract paragraphs of text from their main content, as detected by a method based on (Kohlschütter, Fankhauser, and Nejdl 2010).

In addition to the candidate answers themselves, CRQA extracts certain meta-data, that helps to estimate the relevance of a passage to the current question. For regular web page paragraphs, it is useful to know the topic of the page (*e.g.,* its title) and the context (such as text that immediately precedes the paragraph in the document). For CQA answers, our system stores the text of the corresponding question title, body and category. For convenience, we will refer to this question title and web page title as *"answer topic"*, while the body of the retrieved question and the preceding text block for web candidates as *"answer context"*. Next, for each candidate answer we compute a set of features, described in Table 1.

At the final stage of the module, answers are ranked by their predicted quality. We chose to use the LambdaMART learning to rank algorithm, which was proven to be very successful for various ranking problems (Burges 2010). This model was trained using the RankLib library[5] on the data from last year TREC LiveQA task[6], which includes 1087 questions with answers provided by the participants, each of which was rated on a scale from 1(bad) to 4(excellent) by professional NIST assessors. In a fully automatic setup the top ranked candidate is returned as the final answer to the question.

Answer statistics
— Length in chars, words and sentences — Average number of words per sentence — Fraction of non-alphanumeric characters — Number of question marks — Number of verbs
Answer source
— Binary feature for each of the search verticals: Web, Yahoo! Answers, Answers.com, WikiHow.com
N-gram matches
— Cosine similarities using uni-, bi- and tri-gram representations of the question title and/or body, and answer text, topic or context — The lengths of longest spans of matched terms between question title and/or body, and answer text, topic or context
Information Retrieval score
— BM25 scores between question title and/or body, and answer text, topic or context

Table 1: The list of candidate answer ranking features used by the automatic module of our CRQA system

Crowdsourcing module

Unfortunately, fully automatic QA systems still struggle with many difficult questions (Agichtein et al. 2015), therefore we decided to explore crowdsourcing as one of the ways to help the system to deal with these questions. Instead of immediately returning the answers, CRQA sends questions and top-7 ranked candidates to crowd workers and waits for the responses. We chose to give 7 answers based on the average number of rated answers in our preliminary studies (Savenkov, Weitzner, and Agichtein 2016). Since in TREC LiveQA systems had only 60 seconds to answer each

[4]https://datamarket.azure.com/dataset/bing/searchweb
[5]https://sourceforge.net/p/lemur/wiki/RankLib/
[6]https://sites.google.com/site/trecliveqa2016/liveqa-qrels-2015

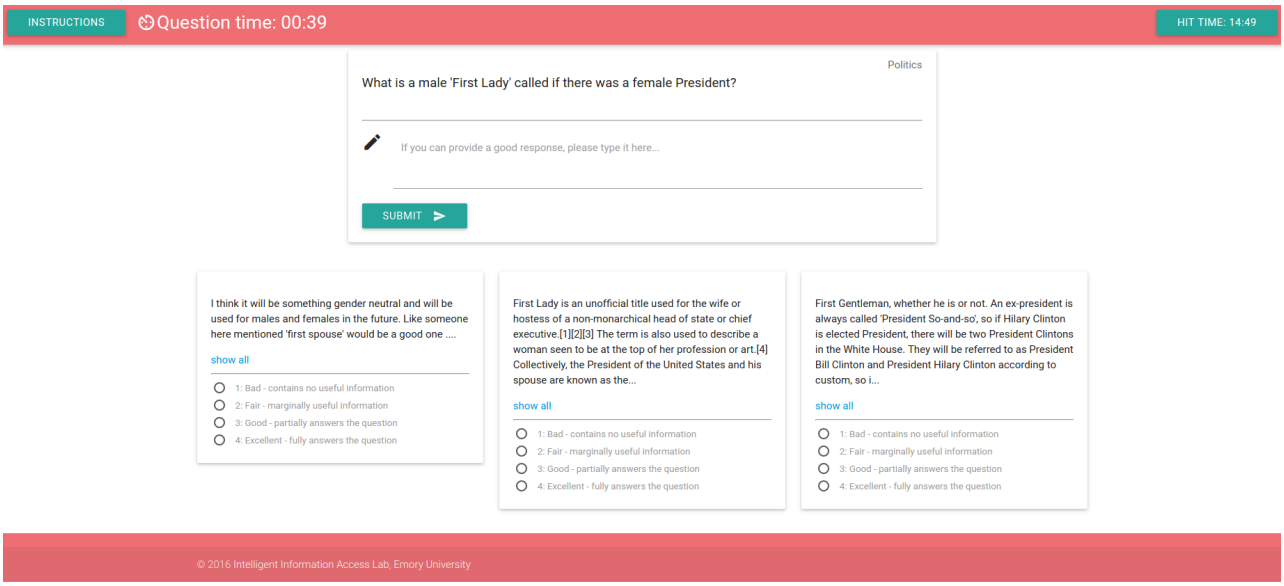


Figure 3: User Interface for workers in our Crowd-Powered Question Answering system

question, we start a timer when a question arrives, and the system waits to receive all worker contributions until the timer reaches 50 seconds to leave some time to generate the final answer. To help resolve the cases when the automatic QA module was not able to generate good candidates, we ask workers to provide their own answers, if possible. Additionally, workers are expected to rate the provided answers, which should help to improve the ranking. Figure 3 presents the user interface of our crowdsourcing module.

The overall algorithm for obtaining crowd input for real-time question answering is the following:

1. When a system receives a question, it is posted to workers, who have 50 seconds to provide their input
2. Workers are asked to write an answer if they can provide one (optional)
3. Otherwise they need to wait for answer candidates to appear
4. When a system is done generating and ranking candidates, it posts top-7 answers for rating (which usually happens $\sim$ 15 seconds after the question is posted)
5. Workers receive a list of answers and rate them until the timer expires. Answers, provided by the workers, are also rated by other workers. Each answer is rated on a scale from 1 to 4, using the official TREC LiveQA rating scale:
   - 1 – Bad: contains no useful information
   - 2 – Fair: marginally useful information
   - 3 – Good: partially answers the question
   - 4 – Excellent: fully answers the question
6. The worker interface displays 3 answers at a time, and when an answer gets rated, it disappears and its place is taken by another answer from the pool. The interface displays only the first 300 characters of the answer, which was experimentally shown to be enough on average to make a good judgment. Full answer can be revealed upon clicking the "show all" link.
7. When the question expires, it disappears, and workers wait for the next question

To hire the workers we used Amazon Mechanical Turk platform[7]. Since the challenge was to run the system "live" over the period of 24 hours, we adapted the retainer model (Bernstein et al. 2011; Bigham et al. 2010), *i.e.,* workers were paid to stay on our interface and complete the tasks for 15 minutes. Specifically, to obtain an even distribution of workers over the 24-hour period of the TREC LiveQA shared task, we posted 10 tasks every 15 minutes. Since not all assignments were accepted right away, the number of workers for each question varied and could be greater than 10. When a worker first gets to our crowdsourcing interface, she is shown task instructions (Table 2) and asked to wait for the questions to arrive. The workers were paid $1.00 for a 15 minutes task, no matter how many questions they received.

We should note, that the setup of TREC LiveQA shared task was favorable for the retainer crowdsourcing model, as the questions were arriving almost every minute uniformly over 24 hour period, which diminishes the waiting and worker idleness problems (Lasecki et al. 2013). An interesting challenge, that we leave for future work, is how to adjust such a crowd-powered QA system to varying questions volume, while optimizing the costs. One idea is to allocate crowdsourcing resources to the incoming questions proportionally to the expected effect, *i.e.,* order the questions by the expected performance of the automatically generated answer. There are also certain techniques to optimize the costs of the retainer model (Bernstein et al. 2012a). In addition, it is also possible to reduce the latency associated with hiring new workers on crowdsourcing platforms on demand, *e.g.,*

[7] http://mturk.com

using quikTurkIt approach from (Bigham et al. 2010).

Instructions
1. This HIT will last exactly 15 minutes 2. Your HIT will only be submitted after these 15 min. 3. In this period of time you will receive some questions, that came from real users on the Internet 4. Each question has a time limit after which it will disappear and you will need to want for the next one 5. If you know the answer to the question, please type it in the corresponding box 6. At some point several candidate answers will appear at the bottom of the page 7. Please rate them from 1 (bad) to 4 (excellent) 8. Do not close the browser or reload the page as this will reset your assignment.

Table 2: Crowdsourcing task instructions, displayed to the user when she first gets to the task

Answer re-ranking and selection

The last stage in CRQA is answer re-ranking, which aggregates all the information received from the crowdsourcing and produces the final answer to the question. The input of the re-ranking module is a set of candidate answers with quality ratings provided by the crowd workers. During the TREC LiveQA 2016 run we used a simple heuristic model, which ordered the answers by the average rating, and returned either the top candidate, if its average score was ≥ 2.5, or the longest worker contributed answer. After the challenge was over, we collected all the questions and answers, that were shown to the crowd workers. The quality of each candidate answer was judged using traditional batch-mode crowdsourcing on the above mentioned scale from 1 (bad) to 4 (excellent). Then, on a subset of the questions along with worker contributions, we trained a new model to re-rank the answers given all available crowdsourcing information. The rest of the questions from TREC LiveQA 2016 run were used to evaluate the quality of the model. This time, for convenience, we used Gradient Boosting Regression Trees (Friedman 2002) from scikit-learn Python package, and trained a regression model to predict the quality of the answer candidates. The features, used for answer re-ranking are listed in Table 3.

Answer-based
— The length of the answer — Source of the answer (Crowd, Web, Yahoo! Answers, Answers.com or WikiHow.com) — Original rank of the candidate answer or -1 for answers provided by the crowd workers
Worker ratings
— Number of ratings provided — Minimum, maximum, median and average ratings

Table 3: The list of features used for answer re-ranking based on crowdsourcing input

Experiments

Experimental Setup: TREC LiveQA

The experimental evaluation of our CRQA system was done on data from the official run of TREC LiveQA 2016 shared task, which happened on May 31, 2016. All participating systems were running for 24 hours and received questions sampled from the live (real-time) stream of questions, posted to Yahoo! Answers. In total, each system received 1,088 questions, and responses were recorded by the organizers. Overall statistics are provided in Table 4. As we can see, on average, workers were able to provide at least one answer to each question, and 6 ratings for each answer.

Name	Value
Number of questions received	1088
Number of completed 15 min assignments	889
Avg number of questions per assignment	11.44
Total cost per question	$0.81
Avg number of answers provided by workers	1.25
Avg number of ratings per answer	6.25

Table 4: Aggregate statistics of CRQA crowdsourcing tasks on TREC LiveQA 2016

Answer Quality Evaluation

We collected all the questions along with system candidates from the official TREC LiveQA 2016 run. In addition, on June 2, two days after the TREC LiveQA challenge has completed, we crawled community answers for all task questions. To obtain the quality labels for these answers and answer candidates, that were shown to crowd workers during the task, we used traditional (batch-mode) crowdsourcing. All the answers were randomly shuffled and rated on a scale from 1 (bad) to 4 (excellent) by workers hired on Amazon Mechanical Turk[8]. Existing research demonstrated, that such crowdsourced labels correlates well with the official ratings, provided by the professional NIST assessors (Savenkov, Weitzner, and Agichtein 2016). Each answer was labeled by 3 different workers, and we averaged the scores to get the final quality labels for the candidates. As mentioned previously, we split the questions multiple times randomly into the training set to build a re-ranking model, and used the other split to evaluate the quality of the model.

Methods compared. We compared CRQA system against several baselines:

- *Automatic QA*: fully automatic QA system
- *CRQA*: automatic QA system with the crowdsourcing module and trained re-ranking model
- *Re-ranking by score*: a simplified version of CRQA re-ranking model, which selects the answer with the highest average ratings, provided by the crowd workers during the task.

[8]There was no time limit for rating this time

- *Yahoo Answers*: traditional, non-real-time community question answering site (Yahoo! Answers), from which the challenge question originated. The answers were collected two days after the challenge, thus allowing the Yahoo Answers community extra two days to collect the answers.

Metrics. To evaluate the methods we used the metrics proposed by the organizers of the LiveQA task:

- **avg-score**: average score over all questions (missing answers receive a score of 0)
- **avg-prec**: average score over all answered questions
- **succ@i+**: the fraction of answers with score i or greater (i=2..4)
- **prec@i+**: the number of answers with score i or greater (i=2..4) divided by the number of answered questions[9]

Table 5 summarizes the performance of the baselines and our system. As we can see, the average score and precision of answers generated by CRQA system is higher than the baseline ranking and even community answers on the Yahoo! Answers platform. However, community answers have higher percentage of *"4 (excellent)"* scores. Figure 4 shows the distribution of scores for the original system ranking, our crowdsourcing system and Yahoo! Answers. Two peaks on the distribution of scores from Yahoo! Answers community suggest, that there are essentially two kinds of responses: non-useful (*e.g.,* spam) or excellent that fully answers the question. In addition, around 20% of the questions did not get any answer from the community. Automatically generated answers, on the contrary, are rarely empty, but on average provide only marginally relevant information, which often does not answer the questions, and therefore rated *"2 (fair)"*. The introduction of the crowdsourcing module allowed CRQA to cover couple percents of the questions, for which the automatic system was not able to generate any candidates, as well as select better candidates when it was possible using crowd ratings.

Therefore, we can conclude, that crowdsourcing can effectively help the automatic QA system to improve the performance of question answering, by providing worker generated answers and rating existing candidates. To get some insights into when crowdsourcing is more and less effective, we looked into the quality of the generated answers across different Yahoo! Answers categories. Overall, CRQA improves upon the baseline fully automatic QA system across all the categories. However, the absolute value of the average quality gain is different, *e.g.,* crowdsourcing had the biggest impact on the quality for the "Travel" category, which was one of the hardest for the fully automatic system (Savenkov 2015). On the other hand, for "Arts & Humanities", "Pets" and "Home & Garden" categories crowdsourcing was less efficient (but still improved the quality). One of the reasons is that questions from these categories often require certain expertise or prior experience, which makes it harder for crowd workers to contribute. To study

[9]Since for each answer we averaged 3 ratings by different workers, the number of answers with the average score of 4 is low

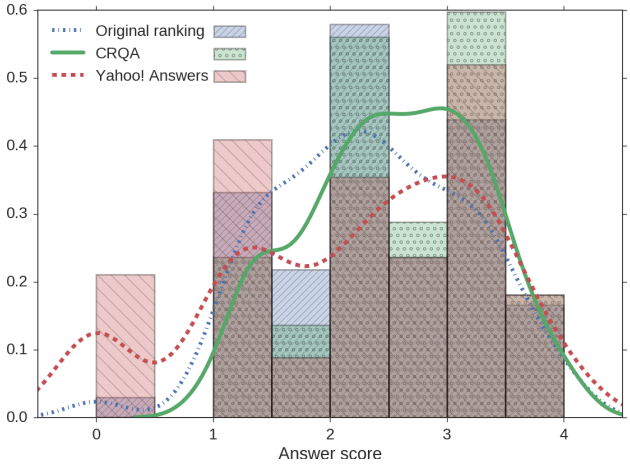


Figure 4: Histogram and kernel density estimation of answer scores for original candidate ranking, CRQA model re-ranking and Yahoo! Answers answers

this hypothesis we further sampled questions with lowest and highest differences in answer quality between the automatic system and CRQA. Many of the questions, where crowdsourcing actually hurt the performance required some special knowledge, *e.g.,* "*Can someone help answer questions about scapholunate ligament surgery rehabilitation?*" or "*What was pol pot's vision of an agrarian society?* ". Examples, when crowdsourcing helped the most, include cases of automatic system failures, *e.g.,* no answer candidates or candidate, that only repeat the question and was ranked first by the system.

To summarize, our Crowd-powered Real-time Question Answering system substantially improves the quality compared to the baseline fully automatic system, and generated answers are often even preferred to responses posted by CQA community.

Analysis and Discussion

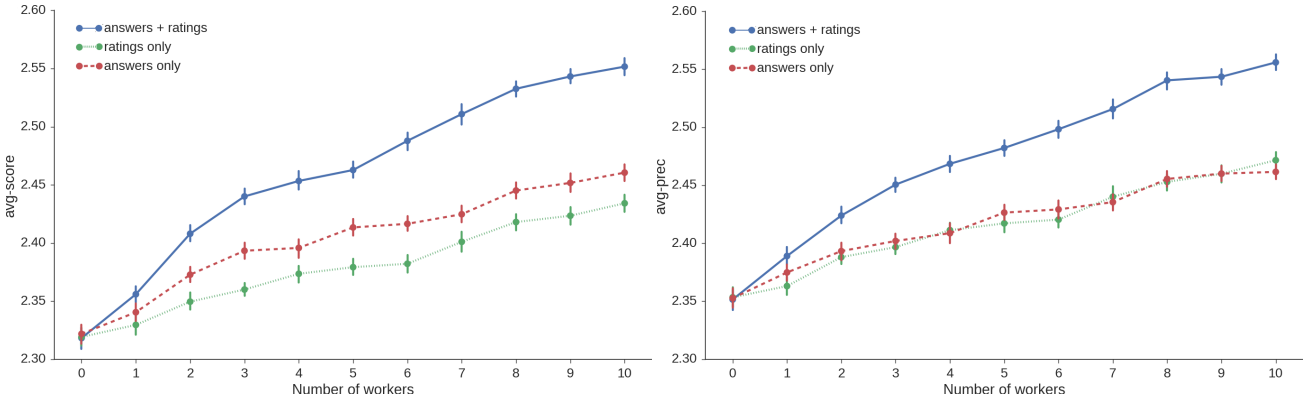
In this section we will analyze some of the results of our experiments and discuss their implications.

Worker answers vs ratings

First, let's look at the contribution of additional answers and ratings provided by the workers. These two types of contributions are complimentary to each other and attempt to solve different problems. Table 5 shows the performance of our question answering system using each of these types of feedback independently. The results demonstrate that both answers and ratings have positive effect on the performance. Even with limited time, workers were able to reliably rate candidate answers, which helped the system to select a better final answer and improve the model precision. However, this method does not help the system in cases, when it was not able to generate any good candidates in the first place, therefore using ratings only has lower average answer score than using worker generated answers. By asking the crowd to provide a response if they can answer the question, CRQA

Method	avg-score	avg-prec	succ@2+	succ@3+	succ@4+	prec@2+	prec@3+	prec@4+
Automatic QA	2.321	2.357	0.697	0.297	0.026	0.708	0.302	0.026
Re-ranking by score	2.416	2.421	0.745	0.319	0.031	0.747	0.320	0.031
Yahoo! Answers	2.229	2.503	0.656	0.375	**0.045**	0.737	**0.421**	**0.050**
CRQA (ratings + ans.)	**2.550**	**2.556**	**0.799**	**0.402**	0.034	**0.800**	0.402	0.034
worker ratings only	2.432	2.470	0.750	0.348	0.030	0.762	0.354	0.031
worker answers only	2.459	2.463	0.759	0.354	0.029	0.760	0.355	0.029

Table 5: Evaluation of the baselines and system answers quality based on the ratings of answers obtained via crowdsourcing. The scores are averaged over 100 different 50:50 splits of 1088 questions into the training and test set. The differences between average score and precision of CRQA and the original ranking are significant at p-value < 0.01



(a) avg-score: Average score per question

(b) avg-prec: Average score per answer (ignoring non-answered questions)

Figure 5: Plot showing how the quality of the final answer depends on the number of workers per question

covers this gap, which is important as in a real scenario even a fair answer would probably be better for the user than no answer at all. Of course, given limited time and the fact that a random worker might not possess an expertise required, such answers do not always perfectly answer the question. Table 6 gives some examples of worker generated answers with low and high quality scores.

To summarize, ratings of answer candidates and worker generated answers both have similar positive effect on the performance of our question answering system. What is more important, the contributions are independent and therefore it is beneficial to use both of them in the final system.

Selection of answer candidate for rating

Predicting the quality of answers and ranking them to select the best is quite challenging for automatic question answering (Surdeanu, Ciaramita, and Zaragoza 2011). External feedback, such as noisy answer ratings, obtained from the crowd workers, provides valuable information, which, as our results demonstrate, can help a QA system to better re-rank the answers. However, the capacity of crowdsourcing for answer ratings is limited, as systems often deal with hundreds and thousands of answer candidates for a given question. In this work, we chose to rate only top-7 answers according the automatic system ranking. This decision was made based on the average number of ratings workers could provide in the allotted time. However, the order in which the answers are shown can also have a strong effect on the system performance, because the answers are typically rated one by one in the order they are displayed on the screen. To study the effect of the answer presentation order on the final answer quality, our system implemented two different strategies, which were selected at random for each question and each worker. The first strategy ordered the answers according to their predicted relevance score, and the other shuffled the answers randomly. The later strategy provides a uniform coverage for all the answers selected for rating, while the former puts more emphasis on the currently top scoring candidates. To analyze the performance of each of the strategies we compute averages score of answers, generated using the corresponding ratings. The average score for answers generated when candidates are shuffled is *2.508*, and it is *2.539* when the candidates are sorted according to their model ranking score. This suggests, that it is beneficial to allocate more of the workers attention on the top scoring candidate answers.

Cost analysis

In this section we analyze the costs of crowdsourcing for real-time question answering. In our study we paid workers $1.00 per single 15 minutes task, where we expected to receive 15 questions. Each 15 minutes we had 10 assignments for different workers, which translates to $15.00 per 15 minutes. Since not all assignments were accepted, overall, our

Question	Answer	Score
Is Gotu Kola a good herb for mental health? How long does it take to work??	yes	1.66
Can I write any number on line number 5 of a W2? would like to set up my W2 were I get the most out of my paycheck and not have to pay taxes at the end of the year...	W2	1.33
I need help with my mum? Something traumatic happened to me about 4 years ago i randomly asked my mother why when I lived with you in your home country a man that was our neighbour used to call me his daughter and the younger kids that lived there called me there cousins and one boy called me his sister?	yes	1.0
Is it bad not wanting to visit your family?	It's nt bad. Just be honest with them. They may be upset but they should understand	3.0
Any health concerns with whey protein? So I workout 3-5 days a week and i drink a whey protein isolate after each workout. Since I workout almost everyday, is it ok for me to just drink a shake everyday?..	As long as you use it as directed, there should not be any major problems. You may want to consult your doctor just in case, but I would not be too concerned.	3.0
Foot pain unable to walk? Hi so today woke with some pain, I'm able to put weight on my heel with no problem or pain. But the area between my heel and toes hurts really bad when I try to go with the motion of taking a step. Its not swollen and I do not remember hurting it at all	Possible gout in your foot, also possible you may have strained it during the previous day.	3.0
What is a good remedy/medicine for stomach aches? Specifically ones caused by stress or anxiety?	Chamomile tea should help	3.66

Table 6: Examples of answers provided by the crowd workers and their average quality scores

experiment cost $0.88 per question, and in this section we will discuss some ideas to reduce it.

First, we will study the effect of the number of workers on the performance of our CRQA system. For this experiment we randomly sampled certain percentage of workers and removed all contributions (answers and ratings) of others. Figure 5 plots the dependency of the performance of our QA system on the number of workers.

Intuitively, more workers mean more reliable answer ratings and more answer candidates, which improves the performance of the question answering system. However, we can observe diminishing returns, the cost per extra gain in performance metrics decreases as the number of workers grows. Half of the overall performance improvements could be achieved with only 3 workers per question, which would save $\sim$70% of the costs.

An alternative cost-reduction strategy is *selective* crowdsourcing, which would only ask for workers feedback for some of the questions. Such a strategy would be necessary to scale a crowd-powered question answering system to a higher volume of questions. There are multiple different approaches for such selective crowdsourcing: *e.g.,* a system can only ask for crowd contributions if it did not generate enough candidate answers or the predicted quality of the top scoring candidates was low (Carmel and Yom-Tov 2010; He and Ounis 2006). We leave these questions for future work, as here we focused on the scenario, proposed by the organizers of the TREC LiveQA shared tasks, where questions arrive one by one and it is possible to utilize crowd input for every question.

To summarize, in the explored real-time QA scenario it is possible to reduce the costs of crowdsourcing by reducing the number of workers, although with some performance losses. Our analysis suggests that paying $\sim$30% of the original cost would give $\sim$50% of the performance improvements.

Related Work

Using the wisdom of a crowd to help users satisfy their information needs has been studied before in the literature. For example, offline crowdsourcing can be used to prepare answers to tail search queries (Bernstein et al. 2012b). In this work, log mining techniques were used to identify potential question-answer pairs, which were then processed by the crowd to generate the final answer. This offline procedure allows a search engine to increase the coverage of direct answers to user questions. In our work, however, the focus is on online question answering, which requires fast responses to the user, who is unlikely to wait more than a minute. Another related work is targeting a different domain, namely SQL queries. The CrowdDB system of (Franklin et al. 2011) is an SQL-like processing system for queries, that cannot be answered by machines only. In CrowdDB human input is used to collect missing data, perform computationally difficult functions or matching against the query. In (Aydin et al. 2014) authors explored efficient ways to combine human input for multiple choice questions from the "Who wants to be a millionaire?" TV show. In this scenario going with the majority for complex questions is not effective, and certain answerer confidence weighting schemas can improve the re-

sults. CrowdSearcher platform of (Bozzon, Brambilla, and Ceri 2012) proposes to use crowds as a data source in the search process, which connects a searcher with the information available through the users of multiple different social platforms. In general, such websites open up many opportunities to interact with their users, in particular, identify users who might possess certain knowledge and request it by asking questions. For example, (Nichols and Kang 2012; Nichols et al. 2013; Mahmud et al. 2013) showed that it is possible to get the information about airport security waiting times or product reviews by posting questions to a social network users, who identified themselves as being at an airport or mentioned the product of interest correspondingly. While in this work, we primarily focused on more traditional way of hiring the crowd workers using Amazon Mechanical Turk, integration with social services is an interesting direction for the future work.

Many works have used crowdsourcing to get a valuable information that could guide an automatic system for some complex tasks. For example, entity resolution system of (Whang, Lofgren, and Garcia-Molina 2013) asks questions to crowd workers to improve the results accuracy. Using crowdsourcing for relevance judgments has been studied extensively in the information retrieval community, *e.g.,* (Alonso, Rose, and Stewart 2008; Alonso and Baeza-Yates 2011; Grady and Lease 2010) to name a few. The focus in these works is on document relevance, and the quality of judgments crowdsourced offline. Whereas in our paper we are investigating the ability of a crowd to quickly assess the quality of the answers in a nearly real-time setting. The use of crowdsourcing in IR is not limited to relevance judgments. The work of (Harris and Srinivasan 2013) explores crowdsourcing for query formulation task, which could also be used inside an IR-based question answering system. (Lease and Yilmaz 2013) provides a good overview of different applications of crowdsourcing in information retrieval.

Crowdsourcing is usually associated with offline data collection, which requires significant amount of time. Its application to (near) real-time scenarios poses certain additional challenges. (Bernstein et al. 2011) introduced the retainer model for recruiting synchronous crowds for interactive real-time tasks and showed their effectiveness on the best single image and creative generation tasks. VizWiz mobile application of (Bigham et al. 2010) uses a similar strategy to quickly answer visual questions. Our work builds on these ideas and uses the retainer model to integrate a crowd into a real-time question answering system. The work of (Lasecki et al. 2013) showed how multiple workers can sit behind a conversational agent named Chorus. Similarly to our work, Chorus used crowd workers to propose and vote on responses to user messages. However, our CRQA system represents a hybrid approach to question answering, where the automatic QA module proposes certain answer candidates, and workers can judge their quality as well as propose additional responses. Such an approach allows us to focus on more complex informational questions, for many of which the workers might not know the answer, but still can contribute by estimating the quality of automatically generated candidates. Chorus, on the contrary, considered somewhat simpler tasks (*e.g.,* things to do in a new place, dinner date ideas, *etc.*), but focused more on maintaining a coherent dialog, which poses additional challenges, such as building a working dialog memory, keeping the users engaged with the dialog using gamification. These ideas can be combined with the ideas of our work to build more intelligent assistants, that do not rely completely on the expertise of the workers. Another use of a crowd for maintaining a dialog is presented in (Bessho, Harada, and Kuniyoshi 2012), who let the crowd handle difficult cases, when a system was not able to automatically retrieve a good response from the database of twitter data.

Conclusions and Future Work

In this paper we presented CRQA, the first, as far as we know, real-time question answering system that integrated crowd work within an automatic QA system. Specifically, we explore different methods of obtaining input from the crowd, and use a machine-learned answer re-ranking model that incorporates the crowd input as features to select the final system answer to return to the user.

We report a large-scale experiment, in which over a thousand real user questions were submitted to the CRQA system in real time, as part of the LiveQA 2016 challenge. CRQA was able to successfully answer these questions in under 1 minute, with over 80% of the answers subsequently rated to be fair or better by human judges. Importantly, CRQA significantly improved question quality and coverage compared to the starting automatic system, and, surprisingly, was often able to return a better answer, compared to the traditional CQA system with millions of users (Yahoo! Answers) with answers collected more than *two days* after the original posting time.

The described CRQA implementation is a promising step towards efficient and close integration of crowd work and automatic analysis for real-time question answering. It raises many promising issues and opens directions for future work, such as the performance of the crowdsourcing module when given less or more time to work, selective crowdsourcing for only the questions deemed "difficult" for the automatic system; more efficient online learning for obtaining ratings from the crowd and integrating them into the ranking model; and investigating additional features and sources of evidence for improving the joint ranking of the system and crowd input. This paper provides a flexible and powerful framework for combining the powers of crowdsourcing with automatic question answering techniques, for building the next generation of real-time question answering systems.

Acknowledgments

The authors thank Scott Weitzner and Alec Wolyniec for helping with some experiments and contributing to the code used in system implementation. The authors are also grateful to the anonymous reviewers for their valuable comments and suggestions. This work was partially supported by the Yahoo Labs Faculty Research Engagement Program (FREP).

References

Agichtein, E.; Carmel, D.; Harman, D.; Pelleg, D.; and Pinter, Y. 2015. Overview of the trec 2015 liveqa track. In *Proceedings of TREC 2015*.

Alonso, O., and Baeza-Yates, R. 2011. Design and implementation of relevance assessments using crowdsourcing. In *Advances in information retrieval*. Springer. 153–164.

Alonso, O.; Rose, D. E.; and Stewart, B. 2008. Crowdsourcing for relevance evaluation. *SIGIR Forum* 42(2):9–15.

Aydin, B. I.; Yilmaz, Y. S.; Li, Y.; Li, Q.; Gao, J.; and Demirbas, M. 2014. Crowdsourcing for multiple-choice question answering. In *AAAI*, 2946–2953.

Bernstein, M. S.; Brandt, J.; Miller, R. C.; and Karger, D. R. 2011. Crowds in two seconds: Enabling realtime crowd-powered interfaces. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, 33–42. ACM.

Bernstein, M. S.; Karger, D. R.; Miller, R. C.; and Brandt, J. 2012a. Analytic methods for optimizing realtime crowdsourcing. *arXiv preprint arXiv:1204.2995*.

Bernstein, M. S.; Teevan, J.; Dumais, S.; Liebling, D.; and Horvitz, E. 2012b. Direct answers for search queries in the long tail. In *Proceedings of the SIGCHI conference*, 237–246.

Bessho, F.; Harada, T.; and Kuniyoshi, Y. 2012. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *Proceedings of SIGDIAL*, 227–231. Stroudsburg, PA, USA: Association for Computational Linguistics.

Bigham, J. P.; Jayant, C.; Ji, H.; Little, G.; Miller, A.; Miller, R. C.; Miller, R.; Tatarowicz, A.; White, B.; White, S.; et al. 2010. Vizwiz: nearly real-time answers to visual questions. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, 333–342.

Bozzon, A.; Brambilla, M.; and Ceri, S. 2012. Answering search queries with crowdsearcher. In *Proceedings of WWW*, WWW '12, 1009–1018.

Burges, C. J. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning* 11:23–581.

Carmel, D., and Yom-Tov, E. 2010. Estimating the query difficulty for information retrieval. *Synthesis Lectures on Information Concepts, Retrieval, and Services* 2(1):1–89.

Carmel, D.; Shtalhaim, M.; and Soffer, A. 2000. eresponder: Electronic question responder. In *International Conference on Cooperative Information Systems*, 150–161. Springer.

Franklin, M. J.; Kossmann, D.; Kraska, T.; Ramesh, S.; and Xin, R. 2011. Crowddb: answering queries with crowdsourcing. In *Proceedings of the 2011 ACM SIGMOD International Conference*, 61–72.

Friedman, J. H. 2002. Stochastic gradient boosting. *Computational Statistics & Data Analysis* 38(4):367–378.

Grady, C., and Lease, M. 2010. Crowdsourcing document relevance assessment with mechanical turk. In *Proceedings of the NAACL HLT 2010 workshop on creating speech and language data with Amazon's mechanical turk*, 172–179. Association for Computational Linguistics.

Harris, C. G., and Srinivasan, P. 2013. Comparing crowd-based, game-based, and machine-based approaches in initial query and query refinement tasks. In *Advances in Information Retrieval*. Springer. 495–506.

He, B., and Ounis, I. 2006. Query performance prediction. *Information Systems* 31(7):585–594.

Kohlschütter, C.; Fankhauser, P.; and Nejdl, W. 2010. Boilerplate detection using shallow text features. In *Proceedings of the Third ACM WSDM Conference*, WSDM '10, 441–450.

Lasecki, W. S.; Wesley, R.; Nichols, J.; Kulkarni, A.; Allen, J. F.; and Bigham, J. P. 2013. Chorus: A crowd-powered conversational assistant. In *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*, UIST '13, 151–162.

Lease, M., and Yilmaz, E. 2013. Crowdsourcing for information retrieval: introduction to the special issue. *Information retrieval* 16(2):91–100.

Mahmud, J.; Zhou, M. X.; Megiddo, N.; Nichols, J.; and Drews, C. 2013. Recommending targeted strangers from whom to solicit information on social media. In *Proceedings of IUI*, 37–48. ACM.

Moldovan, D.; Paşca, M.; Harabagiu, S.; and Surdeanu, M. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM TOIS* 21(2):133–154.

Nichols, J., and Kang, J.-H. 2012. Asking questions of targeted strangers on social networks. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, 999–1002. ACM.

Nichols, J.; Zhou, M.; Yang, H.; Kang, J.-H.; and Sun, X. H. 2013. Analyzing the quality of information solicited from targeted strangers on social media. In *Proceedings of the conference on Computer supported cooperative work*, 967–976. ACM.

Savenkov, D.; Weitzner, S.; and Agichtein, E. 2016. Crowdsourcing for (almost) real-time question answering: Preliminary results. In *2016 NAACL Workshop on Human-Computer Question Answering*.

Savenkov, D. 2015. Ranking answers and web passages for non-factoid question answering: Emory university at trec liveqa.

Shtok, A.; Dror, G.; Maarek, Y.; and Szpektor, I. 2012. Learning from the past: answering new questions with past answers. In *Proceedings of WWW*, 759–768.

Surdeanu, M.; Ciaramita, M.; and Zaragoza, H. 2011. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics* 37(2):351–383.

Whang, S. E.; Lofgren, P.; and Garcia-Molina, H. 2013. Question selection for crowd entity resolution. *Proc. VLDB Endow.* 6(6):349–360.