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

Denis Savenkov
Emory University
dsavenk@emory.edu

Eugene Agichtein
Emory University
eugene@mathcs.emory.edu

Abstract

Modern search engines have made dramatic progress in the answering of questions about facts, such as those that might be retrieved or directly inferred from a knowledge base. However, many other questions that real users ask are more complex, such as requests for opinions 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 automated 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 automated 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 are arguably considered "the next big thing", and a number of startups developing this kind of technology has emerged in Silicon Valley and around the world[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, TREC LiveQA shared task[2] was started in 2015, where automated systems attempt to answer real users' 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.

In this work we explore two ways *crowdsourcing* can be used to help an automated 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, we study if crowd workers can quickly and reliably judge the quality of the proposed answer candidates, and if it is possible to obtain reasonable written answers from the crowd within a limited amount of time. 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 automated 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 par-

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

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

ticipating in the TREC LiveQA 2016 evaluation challenge, which provided us with a realistic evaluation setup.

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.
2. Large scale experiments using CRQA to answer real user questions in a live TREC LiveQA 2016 challenge setting.
3. Extensive analysis of the system performance, focusing on the contributions of crowd input to the performance of the overall system.

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

Not sure where to put research questions. They are not 1-1 with contributions.

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 2016 shared task, which affected some of the system design choices.

A series of TREC LiveQA evaluation campaigns was started in 2015 with the goal to facilitate research in automatic question answering for complex informational needs. In 2015 and 2016 versions of the task participants of the challenge develop a live question answering system, that needs to respond to user questions, which are sampled from Yahoo! Answers community question answering website live stream. Each input question consists of the short question title, body and category (Figure 1). A QA system has to provide an answer of 1000 characters or less in under 1 minute time limit. 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, that includes an automated question answering and crowdsourcing modules. The high level architecture is presented in Figure 2.

The automated 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 automated 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 automated 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 the given questions if they can, and additionally rate the quality of candidate answers, generated by the automated system. Worker contributions are then used by a trained re-ranking model, that selects the best candidate answers using all the information available. The next two sections describe the architectures of the automated and crowdsourcing modules of our system.

Automated question answering module

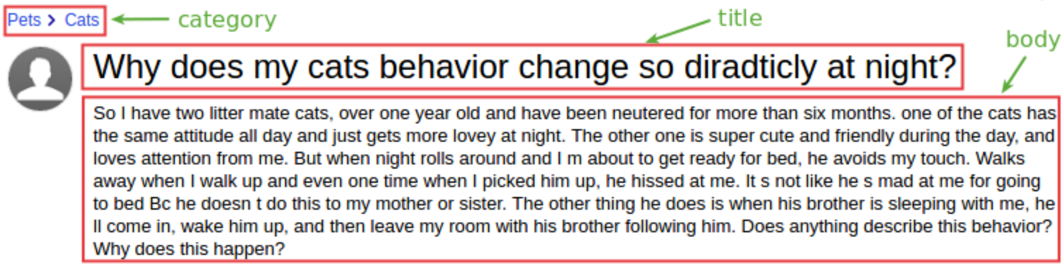
When CRQA receives a question, it generates a set of search queries to retrieve a set of relevant documents and extract candidate answer passages. Search queries are generated using the following strategies:

- 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 (Shtok et al. 2012) 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. Therefore, CRQA uses multiple different CQA data sources, which potentially contain a diverse set of questions. However, quite often it is hard to find a similar question in an archive, and many of the information needs are unique. Therefore, we include a web search component, that can retrieve regular web documents, from which our system can extract candidate answers. More specifically, CRQA queries the web using Bing Web Search API[4], Yahoo! Answers, Answers.com and WikiHow.com using their respective search interfaces. For each query we retrieve top-10 relevant documents and question-answer pairs from CQA archives. As candidate answers CRQA extracts answers to the questions retrieved from CQA results and paragraphs of text from the main content of regular web documents, as detected by a method based on (Kohlschütter, Fankhauser, and Nejdl 2010). Quite often, it's hard to estimate the quality of a candidate answers from its text only. The task becomes much easier if a system has access to additional data, *e.g.* for the candidates generated from the CQA archives, it is useful to know the text of the original question the candidate answers. For candidates, generated from regular web pages, it's useful to know the topic of the page (*e.g.* from its title), additionally the context of the candidate, such as text that immediately precedes it in the document was shown to provide a useful information for QA. Therefore, besides the text of a candidate answer, we keep some potentially useful metadata, *e.g.,* for QnA candidates we include text and category of the retrieved question, and web page title and text block preceding the answer passage in the document for web-search based candidates. For convenience, we will call

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

[4] https://datamarket.azure.com/dataset/bing/searchweb



Why does my cats behavior change so diradticly at night?

So I have two litter mate cats, over one year old and have been neutered for more than six months. one of the cats has the same attitude all day and just gets more lovey at night. The other one is super cute and friendly during the day, and loves attention from me. But when night rolls around and I m about to get ready for bed, he avoids my touch. Walks away when I walk up and even one time when I picked him up, he hissed at me. It s not like he s mad at me for going to bed Bc he doesn t do this to my mother or sister. The other thing he does is when his brother is sleeping with me, he ll come in, wake him up, and then leave my room with his brother following him. Does anything describe this behavior? Why does this happen?

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

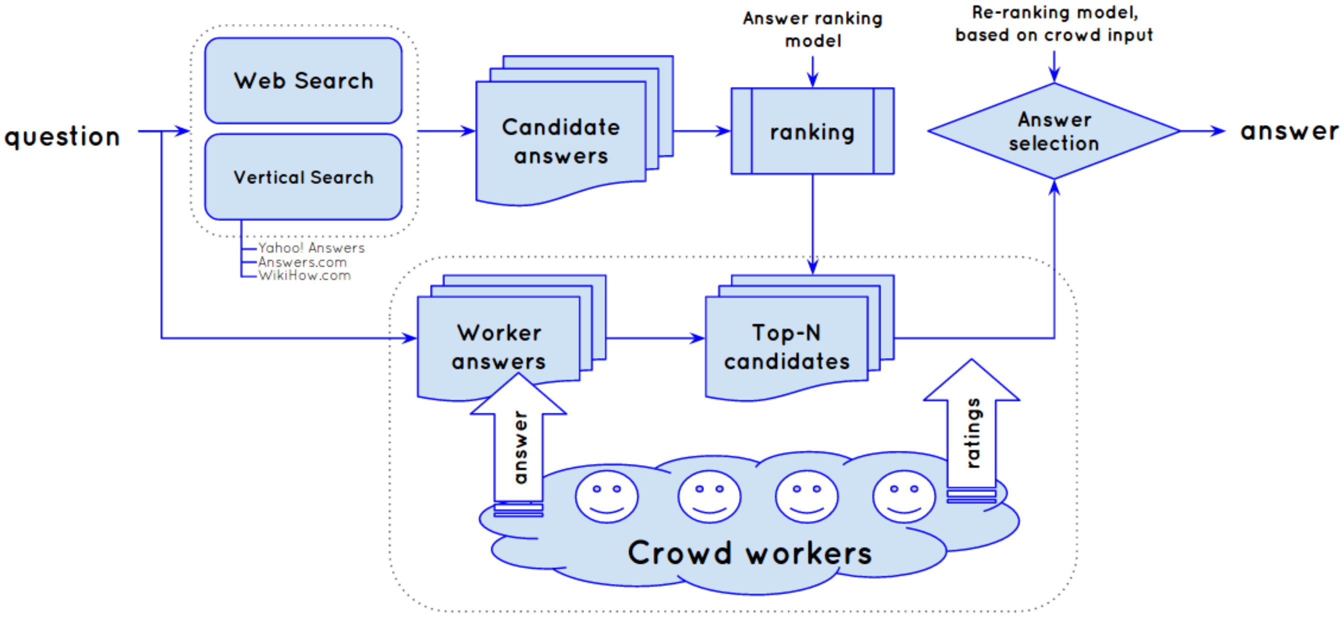


Figure 2: The architecture of our Crowd-powered Real-time Question Answering system, that uses crowdsourcing to augment a list of automatically extracted candidate answers and to rate their quality

the title of a retrieved question or web page *"the answer topic"*, and the body of the retrieved question or the preceding text block the *"the answer context"*.

Next, for each candidate answer we compute a set of features, described in Table 1.

At the final stage of the module, the answer are ranked by their predicted quality. We chose to use the LambdaMART learning to rank algorithm, which was proven to be a very successful choice 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 automated setup the top candidate could be returned as the final answer to the question, but in this work we explore if we can further improve the performance of the system using crowdsourcing.

Crowdsourcing module

The paper might also be improved by including **design principles for crowd powered QA systems**. This might be very helpful for researchers and practitioners.

As we mentioned before, two of the main problems with the fully automated system responses are lack of good candidates and problems with ranking, therefore, we decided to explore if crowdsourcing can be helpful to overcome these challenges in near real-time scenario. Instead of returning the final answer, the system sends the question and top-7 ranked candidates to the 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. Since systems had only 60 seconds to answer each 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 the system some time to generate the final answer and minimize the chances of going above the time limit. CRQA asks workers to write their own answers to

[5] https://sourceforge.net/p/lemur/wiki/RankLib/

[6] https://sites.google.com/site/trecliveqa2016/liveqa-qrels-2015

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 automated module of our CRQA system

questions if possible, which should provide additional candidates in case there are no good automatically generated ones. Additionally, systems expects workers to rate the quality of top-7 ranked candidates, which should help to fix the ranking problems and select a better final answer. 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 the workers, who will have 50 seconds to provide their input
2. Workers are asked to write an answer if they can provide one (it's optional)
3. Otherwise they are waiting for the answer candidates to arrive
4. When a system is done with generating and ranking candidates it posts top-7 scoring answers to the workers for the rating (which usually leaves $\sim$ 35 seconds for rating)
5. Workers receive a list of answers[7] and rate them until the timer runs off. 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 interface displays 3 answers at a time, 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 timer runs off, the question and all the answers disappear, and workers wait for the next question

To hire the workers we used Amazon Mechanical Turk platform[8]. Since the challenge was to run the system "live" over the period of 24 hours, we adapted the retainer model to our question-answering task, inspired by its success in existing research (Bernstein et al. 2011; Bigham et al. 2010). 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, and they expired after the next set of tasks became available. Since not all assignments were accepted by some worker 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 the whole 15 minutes task, no matter how many questions they got[9].

We should note, that the setup of TREC LiveQA shared task was very favorable for this model, as the questions were arriving almost every minute uniformly over 24 hour period, which minimizes the waiting time and worker idleness problem (Lasecki et al. 2013). An interesting challenge, that we leave for future work, is how such a crowd-powered QA system can handle varying questions volume, while optimizing the costs. (Bernstein et al. 2012a) describes a number of techniques to optimize the costs of the retainer model. It's also possible to reduce the latency associated with hiring new workers on a crowdsourcing platform, *e.g.*, quikTurkIt approach of (Bigham et al. 2010). Another idea, applied to question answering, is to prioritize the questions according to their predicted difficulty, *i.e.*, we should allocate more workers to questions, for which the automatic system isn't confident in the generated answer, than to some easy questions.

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. Candidates can include the answers posted by the workers, which might also be rated, if workers had enough time to do that. We trained another learning to rank model[10] to re-rank the answers given all available crowdsourcing information. The re-ranking model was trained on a subset of questions from TREC LiveQA 2016 run[11]. To estimate the

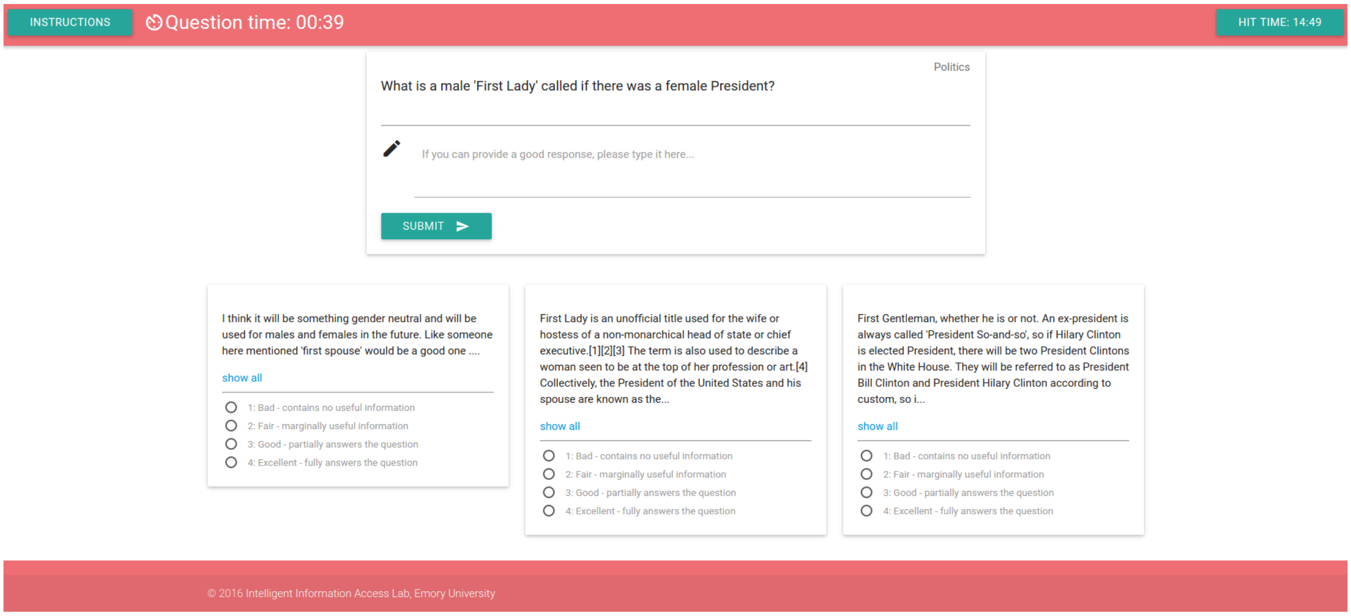
[7] Answers submitted by workers are also sent for ratings to all workers except the author

[8] http://mturk.com

[9] In TREC LiveQA task questions are sent to the systems one by one, therefore there is no concurrency, however the delays between the questions are possible

[10] For convenience, we used Gradient Boosting Regression Trees (Friedman 2002) from scikit-learn Python package.

[11] We should note, that the experiments with trained re-ranking model was conducted after the LiveQA final run. For the shared



What is a male 'First Lady' called if there was a female President?

If you can provide a good response, please type it here...

I think it will be something gender neutral and will be used for males and females in the future. Like someone here mentioned 'first spouse' would be a good one ....

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

First Lady is an unofficial title used for the wife or hostess of a non-monarchical head of state or chief executive.[1][2][3] The term is also used to describe a woman seen to be at the top of her profession or art.[4] Collectively, the President of the United States and his spouse are known as the...

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

First Gentleman, whether he is or not. An ex-president is always called 'President So-and-so', so if Hilary Clinton is elected President, there will be two President Clintons in the White House. They will be referred to as President Bill Clinton and President Hilary Clinton according to custom, so i...

○ 1: Bad - contains no useful information
○ 2: Fair - marginally useful information
○ 3: Good - partially answers the question
○ 4: Excellent - fully answers the question

© 2016 Intelligent Information Access Lab, Emory University

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

Instructions
1. This HIT will last exactly 15 minutes
2. Your HIT will only be submitted after these 15 minutes
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 on a scale 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

quality of answers, that were given to workers for rating, including worker-generated answers, we turned to the traditional batch-model crowdsourcing. Workers were asked to label each answer on a scale from 1-bad to 4-excellent. Then, a subset of the questions was used for training, while the rest was used to estimate the quality of the models.

The features, used for answer re-ranking are listed in Table 3.

CRQA sorts the candidates by the quality score predicted by the model, as returns the top candidate as the final answer.

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

We now describe the experimental setup used to evaluate the performance of CRQA and other methods for near real-time question answering.

Experimental Setup: TREC LiveQA

The experimental evaluation of our CRQA system was done on the official run of TREC LiveQA 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 by real users to Yahoo! Answers platform. In total, each system received 1,088 questions, and system 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 for each question, and each of the provided answers got 6 ratings. Since the interface could show only 3 answers at a time, answers had different chances of being rated. To investigate the effect of the order of the candidates posted for

task we used a simpler heuristic-based re-ranker, described in the next Section

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

Table 4: Aggregate statistics of crowdsourcing tasks

ratings on the quality of the final answer, in CRQA for each worker and each question we randomly selected one of the strategies: ordering answer candidates by their model rank or random shuffling of the candidates. The system variant with shuffled candidates for rating was expected to obtain more diverse and comprehensive set of ratings, as we will investigate in the Analysis section.

Answer Quality Evaluation

The official results of the shared task will be available in November 2016 during the TREC conference[12]. Meanwhile, we used traditional (batch-mode) crowdsourcing to obtain the quality labels for all answer candidates that were given to the workers during the task, as well as the answers provided by the workers. In addition, on June 2, two days after the TREC LiveQA challenge has completed, we crawled the current answers provided by the community for the questions, used for the task. All the answers for each question were randomly shuffled and rated on a scale from 1 (bad) to 4 (excellent) by workers hired on Amazon Mechanical Turk. 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 labelled by 3 different workers, and we averaged the scores to get the final quality labels for the candidates.

Methods compared. We compared CRQA system against several baselines:

- *Automated QA*: automated QA system described in Section .
- *CRQA*: Automated QA system with Crowdsourcing, described in Section
- *Re-ranking by score*: a simplified version of CRQA re-ranking model, which select the answer with the highest average ratings, provided by the crowd workers.
- *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 through traditional (community-based) crowdsourcing.

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

- **avg-score**: average score over all questions
- **avg-prec**: average score over all answered questions
- **succ@i+**: the fraction of answers with score i or greater (i=2..4)
- **p@i+**: the number of answers with score i or greater (i=2..4) divided by the number of answered questions[13]

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, Yahoo! Answers 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 didn't 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 doesn't answer the questions, and therefore rated *"2 (fair)"*. The introduction of the crowdsourcing module allowed CRQA to cover a couple of percent of the questions, for which the automated system wasn't able to generate any candidates, as well as select better candidates when it was possible using crowd ratings.

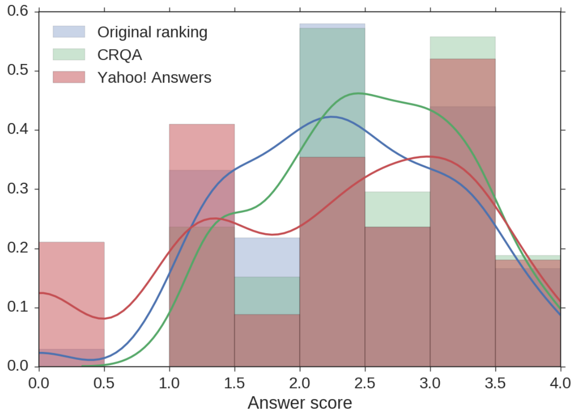


Figure 4: Histogram and kernel density estimation of answer scores for original candidate ranking, CRQA model re-ranking and Yahoo! Answers answers

Therefore, for RQ1 we can conclude, that crowdsourcing can effectively help automated QA system to improve the performance of question answering, by providing worker generated answers and rating existing candidates.

[12] http://trec.nist.gov/

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

Method	avg-score	avg-prec	succ@2+	succ@3+	succ@4+	prec@2+	prec@3+	prec@4+
Automated 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 + answers)	**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

To summarize, our Crowd-powered Real-time Question Answering system substantially improves the quality compared to the baseline fully automated 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 answer ratings provided by the workers. These two types of contributions are complimentary to each other and attempts 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 doesn't help the system in cases, when it wasn't able to generate a good candidate 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 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 to answer the question, such answers don't always perfectly answer the question.

One of the limitations of the paper is that it does not discuss the type of topics / questions on which the approach works well and the type of topics on which the use of crowdsourcing fails.

Table 6 gives some examples of worker generated answers with low and high quality scores.

To summarize our answer to RQ2, 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 one of the main challenges in automated question answering (Surdeanu, Ciaramita, and Zaragoza 2011). External feedback, such as noisy answer ratings, obtained from the crowd workers, provide valuable information, which, as our results demonstrate, can help a QA system to better rerank the answers. However, the capacity of crowdsourcing for answer ratings are obviously limited, as systems often are dealing with hundreds and thousands of answer candidates for a given question. In this work, we made a choice to rate only top-7 answers according the automated system ranking. This decision was made based on the average number of ratings workers could provide in the allotted time[14]. 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. Our system included two strategies for answer ordering: random or according to their ranking score. The former strategy provides a uniform coverage for all the answers selected for rating, while the later puts more emphasis on the currently top scoring candidates. We randomly selected one of the strategies for each user and question. To analyze the performance of each of the strategies we compute the average score of answers, generated using the corresponding ratings. The average score for answers generating when candidates are shuffled is 2.508, and it's 2.539 when the candidates are sorted according to their model ranking score. This suggests, that it's beneficial to allocate more of the workers attention on the top scoring candidate answers.

Cost analysis

The results of our experiments clearly demonstrated that crowdsourcing can improve the performance of near real-time question answering system. The next reasonable question is what is the price of this improvement. In our study we paid workers $1.00 per single 15 minutes task, and each 15 minutes we had 10 assignments, which translates to $15.00 per 15 minutes. Overall, our experiment cost $0.88 per question, and in this section we will discuss some ideas to reduce this cost.

[14] The answers were posted for rating automatically after an automated system was done with candidate generation and ranking. On average users had ~ 35 seconds to provide the ratings.

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 don't 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

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.

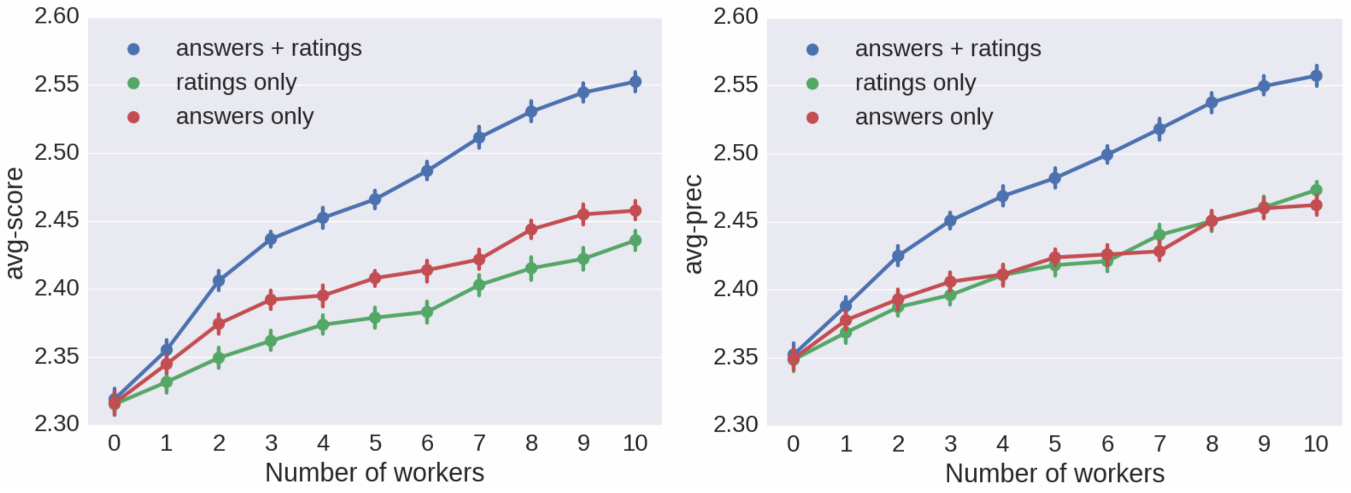
Obviously more workers mean more reliable answer ratings and more answer candidates, which improves the performance of the question answering system. However, we can observe diminishing returns, as the cost per extra gain in performance metrics decreases as the number of workers grow. Half of the overall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

An alternative cost-reduction strategy is selective triggering of 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 didn't generate enough candidate answers or the predicted quality of the top scoring candidates is low (Carmel and Yom-Tov 2010; He and Ounis 2006). We leave this questions for the 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's possible to utilize crowd input for every questions.

To summarize and answer RQ3, 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 30% of the original cost would give 50% of the performance improvement.

Related Work

Using the wisdom of a crowd to help users satisfy their information needs has been studied before in the literature. (Bernstein et al. 2012b) explored the use of crowdsourcing for offline preparation of answers to tail search queries. Log mining techniques were used to identify potential question-answer fragment 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 (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 isn't effective, and certain answerer confidence weighting schemas can improve the re-



(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

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's possible to get the information about airport security waiting times and product reviews by asking 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 automated 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 crowdsourced judgments. 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 isn't limited to relevance judgements. 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. This work builds on these ideas and uses the proposed 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, where human input is used to propose and vote on responses.

Compare with Chorus in more detail. ”how it is different than Chorus or what was missing from such a system like chorus and to what is currently being presented. What was missing in the evaluation that Chorus did vs what you are presenting”

In our work we use similar ideas in application to question answering, which requires more comprehensive responses from 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. In our paper, we focus on a single part of the human-computer dialog, i.e. question answering, which requires a system to provide some useful information in a response to the user.

Conclusion

In this paper we presented CRQA, the first, as far as we know, real-time question answering system that integrated crowd work within an automated question answering system. Specifically, we explore different methods of obtaining input from the crowd, and use a machine-learned answer reranking 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 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. Importantly, CRQA significantly improved question quality and coverage compared to the starting automated-only system, and, surprisingly, was able to return better answers on average 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 automated analysis for real-time question answering. It raises many promising issues and opens directions for future work, such as selective crowdsourcing for only the questions deemed "difficult" for the automated 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 automated question answering techniques, for building the next generation of real-time question answering systems.

Add future work for contextual answers. What if we have a dialog and previous questions matter?

Emphasize, that we are leaving the time analysis for future work. Also, what is the impact of the time limit (60 seconds) on the performance of the system? More precisely, despite the requirement to provide an answer within one minute, what would be the performance of the system given more or less time?

do authors think it is possible to learn from the crowdsourcing rankings and answers to better answer questions automatically? if so, in what way?

Acknowledgments

The authors are 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 *The Twenty-Fourth Text REtrieval Conference (TREC 2015) Proceedings. National Institute of Standards and Technology (NIST).*

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 on human factors in computing systems*, 237–246. ACM.

Bessho, F.; Harada, T.; and Kuniyoshi, Y. 2012. Dialog system using real-time crowdsourcing and twitter large-scale corpus. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, SIGDIAL '12, 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. ACM.

Bozzon, A.; Brambilla, M.; and Ceri, S. 2012. Answering search queries with crowdsearcher. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, 1009–1018. New York, NY, USA: ACM.

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.

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 on Management of data*, 61–72. ACM.

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 International Conference on Web Search and Data Mining*, WSDM '10, 441–450. New York, NY, USA: ACM.

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. New York, NY, USA: ACM.

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 the 2013 international conference on Intelligent user interfaces*, 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 Transactions on Information Systems (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 2013 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 the 21st international conference on World Wide Web*, 759–768. ACM.

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.