Targeted Question Answering on Smartphones Utilizing App Based User Classification

Yavuz Selim Yilmaz Bahadir Ismail Aydin Murat Demirbas
Department of Computer Science and Engineering
SUNY University at Buffalo
Buffalo, New York, 14226, USA
Email: {yavuzsel | bahadiri | demirbas}@buffalo.edu

***Abstract*—State-of-the-art question answering systems are pretty successful on well-formed factual questions, however they fail on the non-factual ones. In order to investigate effective algorithms for answering non-factual questions, we deployed a crowdsourced multiple choice question answering system for playing "Who wants to be a millionaire?" game. To build a crowdsourced super-player for "Who wants to be a millionaire?", we propose an app based user classification approach. We identify the target user groups for a multiple choice question based on the apps installed on their smartphones. Our final algorithm improves the answering accuracy by 10% on overall, and by 35% on harder questions compared to the majority voting. Our results pave the way to build highly accurate crowdsourced question answering systems.**

***Keywords*—targeted crowdsourcing, app-based classification, question answering**

I. Introduction

Question answering (QA) is considered as a fundamental problem by the artificial intelligence (AI) and the machine learning (ML) communities since the early years of the information retrieval (IR) research. Search engines show that computers can answer well-formed factual queries successfully. Still, non-factual questions have been a setback on the overall accuracy of these systems. There has been a substantial research effort on answering this kind of questions by incorporating the IR and the natural language processing (NLP) techniques. Even though there are some encouraging real-life examples [1]–[3], AI systems has a long way to go for answering non-factual questions with a near-perfect accuracy. Crowdsourcing, on the other hand, is a natural fit for QA of non-factual questions because it utilizes the human intelligence to solve this kind of problems which are hard for the computers.

We believe that asking multiple choice questions is more productive than asking open-domain questions for effective crowdsourcing. By definition, crowdsourcing consists of two basic steps: (1) Tasking the crowd with small pieces of a job, and (2) merging the outcomes or the answers in order to complete the entire job. Open-domain questions result in a large set of possible answers which makes it hard to aggregate the responses from the crowd to produce a final answer. On the other hand, presenting binary or multiple choices for the questions facilitates the aggregation. It also makes the job of the crowd easier: punch in the choice (A, B, C, or D) instead of figuring out how to present the answer. This lets the crowd to complete the tasks in a shorter time. We also suggest that providing multiple choice questions is often feasible. The original asker can provide the multiple options when the question is on deciding among some choices (hotels to stay, products to buy, music to listen). Otherwise, it is also possible to automate the process of adding multiple choices to an open-domain question using ontologies and lightweight ML techniques [4].

In this work, we study collaboration techniques for high accuracy multiple choice question answering (MCQA). To this end, we deployed a crowdsourced system to play "Who wants to be a millionaire?" (WWTBAM) [5] live. Our work was inspired by IBM Watson's success at Jeopardy and aims to utilize the crowd to answer WWTBAM questions accurately. We provide an Android app to let the users play the game while the quiz-show is live on TV simultaneously[a]. When the show is on air, this app makes a notification sound to alert the users to pick up their phones and start playing. Our two project members type the questions and multiple choices as they appear on the TV, and the app users enter their answers using their phones. The system has a backend server to dispatch the questions to the app users, and collect the answers from them. We run our MCQA algorithms on this backend server.

To the best of our knowledge, our work is the first crowdsourcing study on WWTBAM game. For the easier questions[b], our majority voting performs over 90% success rate on average. However, on the harder questions[b] the success of the majority voting slips below 50%. Therefore, we study more complex methods to pull the crowd's performance up on the harder questions.

To improve the accuracy on the harder questions, we suggest to define target user groups for those queries by utilizing the applications installed on the smartphones. This classification reduces the number of votes at the group level while increasing the homogeneity of the votes inside the user groups. By this way we are able to identify the appropriate minority voice and design effective MCQA algorithms based on those user groups.

The Android applications in the Google Play Store [6] are categorized into 34 discrete categories. We form 34

[a]The app also has offline question answering option for the users who are not present when the show is on air.

[b]The question difficulty and threshold is determined by the show's format: easier questions are from level 1 to 7, and harder ones are level 8 and above.

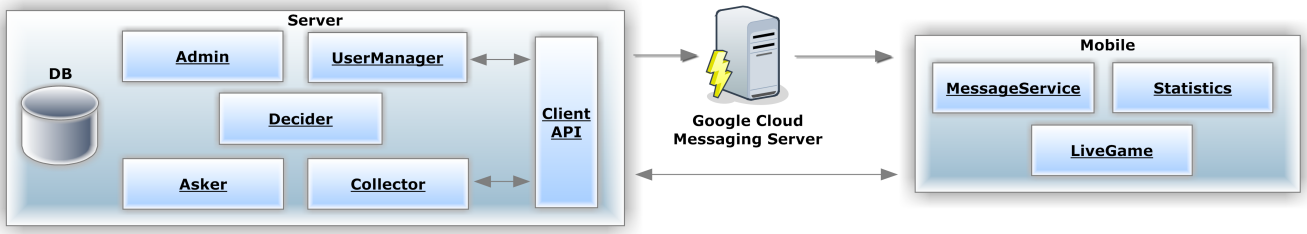


Figure 1. The system architecture

user groups to match those app categories, and then we classify the users into these user groups based on the number of the installed apps from each of those corresponding categories. We then utilize these user groups in our MCQA algorithms to answer the questions accurately. On overall, our final algorithm improves the answering accuracy by 10% compared to the basic majority voting. More importantly, it pulls up the success rate on the harder questions by 35% where majority voting falls short. These results suggest that building a crowdsourced super-player for MCQA using our methods is feasible. In future work, we will investigate adapting lessons learned from the WWTBAM application to general/location-based crowdsourcing applications and recommendation systems.

The rest of the paper is organized as follows: We summarize the design of our system in Section II. Then we provide the details about our dataset in Section III. In Section IV, we present the performance of the majority voting to set a base to our performance evaluations. Then in Section V, we discuss how to build the app based user groups, namely we present our user classification methods. In Section VI, we evaluate the performance of different crowdsourcing algorithms that leverage our user classification methods. Then we conclude the paper with reviewing the related work in Section VII and presenting our future directions in Section VIII.

II. CrowdReply: A Crowdsourced WWTBAM App

We developed an Android app and the backend software to enable the audience watching WWTBAM on the TV to play along on their smartphones simultaneously. We targeted the Turkish audience due to the high popularity of the show there. (Our app has been installed more than 307K times [6].) When the show is on air, CrowdReply app makes a notification sound to alert the users to pick up their phones and start playing. Our two project members type the questions and multiple choices as they appear on the TV, and the app users enter their answers using their phones. The users are incentivized to participate as they enjoy the game-play, and they can see their ranking among the other players. We also provide offline question answering for the users who are not present when the show is on air.

The game enables us to collect large-scale crowdsourcing data about MCQA dynamics. In total, we have more than 5 million answers to more than 3000 questions. The ground-truth of a question is the correct answer announced on the TV. There are up to 12 questions with increasing difficulty levels, and the harder questions appear based on the performance of the TV contestant.

The overall architecture of the CrowdReply is shown in Figure 1. CrowdReply consists of three main parts, an admin part for entering the questions & the multiple choices while the game is live on the TV, a mobile side for presenting the questions to the users and letting them answer the questions, and a server side for dispatching the questions, collecting the answers, and providing useful statistics. We described the design, implementation, and deployment of the CrowdReply in a previous work [7]. In this paper, we leverage this app and the data for targeted MCQA.

III. Our Dataset

In order to define the target for an MCQA query, we leverage app type based user classification by utilizing the applications installed on the smartphones. To study this approach, we collected the installed apps from our users who are currently using the CrowdReply app to play the WWTBAM game [6]. In order to evaluate the feasibility of our approach, we used a subset of our user base. In our dataset, we have 1397 unique devices (i.e. users) and 16651 unique apps installed on them. Figure 2 shows the distribution of the apps over the devices. The graph unveils that, there are about 10 popular apps which are installed on almost every device in our dataset. Furthermore, around 100 apps are installed on 100 devices or more. The remaining apps, which count to more than 16500 apps, are scattered among the devices.

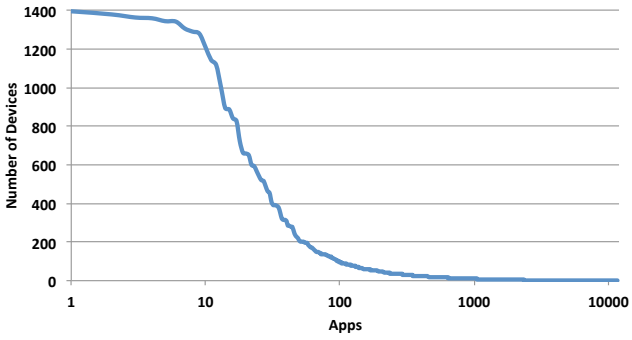


Figure 2. Apps vs the number of devices each app is installed on (app names on the x-axis are replaced with the incremental app id's to make the axis labels readable)

We used Google Play Store [8] listings to categorize the applications. Among those 16651 apps we collected, we were able to categorize 11652 of them. It is due to the fact that

TABLE I
GOOGLE PLAY APP CATEGORIES AND THEIR APPEARANCE IN OUR DATASET

App Category	Num. of Apps	Num. of Devices
Books and Reference	336	690
Business	128	887
Comics	42	75
Communication	214	1389
Education	732	535
Entertainment	990	1324
Finance	121	313
Health and Fitness	162	236
Libraries and Demo	40	250
Lifestyle	403	549
Live Walpaper	0	0
Media and Video	388	1318
Medical	62	76
Music and Audio	467	853
News and Magazines	222	967
Personalization	748	579
Photography	409	724
Productivity	299	1126
Shopping	107	312
Social	202	1339
Sports	198	459
Tools	779	1397
Transportation	77	224
Travel and Local	245	1359
Weather	50	203
Widgets	0	0
Games - Arcade and Action	1126	1145
Games - Brain and Puzzle	1075	1397
Games - Cards and Casino	190	825
Games - Casual	1111	993
Games - Live Wallpaper	0	0
Games - Racing	395	707
Games - Sports Games	334	580
Games - Widgets	0	0

some of the apps are system apps and they don't have a Google Play Store listing, and some others are either never published through Google Play Store or they were removed at the time we crawl the listing data. Table I shows the Google Play Store app categories, the number of unique apps in our dataset, and the number of unique devices which has at least one installed app from the given category.

The format of the WWTBAM game categorizes the questions into 12 levels based on their difficulty. In our experiments, for all 12 question levels[c], we have total of 2654 questions that are answered by our 1397 test users. For these 2654 questions, we have total of 94735 answers. Table II shows the distribution of the questions and the answers by question level.

[c]As there exist no level 12 question in our dataset, we exclude this level from our results hereafter.

TABLE II
TOTAL NUMBER OF QUESTIONS AND ANSWERS BY QUESTION LEVEL IN OUR DATASET

Question Level	Num. of Questions	Num. of Answers
1	508	27742
2	471	20643
3	443	16437
4	343	10766
5	310	9837
6	238	4639
7	176	3080
8	87	994
9	40	386
10	24	139
11	14	72
12	0	0

IV. THE NAIVE APPROACH: MAJORITY VOTING

An MCQA algorithm for our WWTBAM game tries to answer a question using crowds' responses. The success of an algorithm is defined as the percentage of the correctly answered questions in a given question level. For example, if an algorithm is able to answer p number of questions out of total Q questions from level l, then its success S for the question level l would be:

$$S_l = \left(100 \times \frac{p}{Q}\right)\% \quad (1)$$

In order to analyze the success of our classification methods and the accuracy of our MCQA algorithms, we compare the results with our base algorithm: *majority voting*. Majority voting for MCQA works as follows: Given a question and a set of answers, the algorithm counts the user answers for each choice, and then selects the mostly voted choice as the final answer. Figure 3 shows the success of the base majority voting algorithm for our dataset.

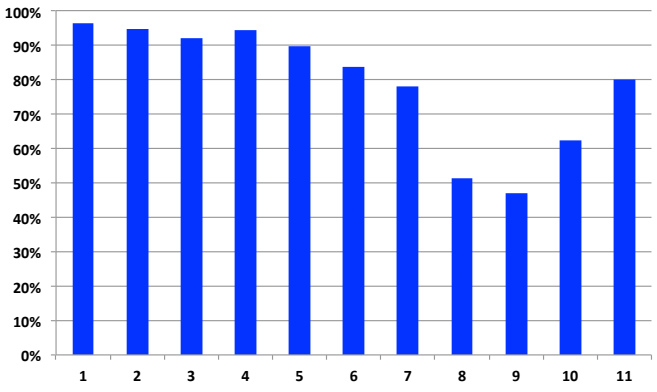


Figure 3. Base majority voting success by question level

While the overall success of the base majority voting algorithm for the easier questions is above 90%, it dramatically

decreases for the harder questions (slips below 50%). Namely, the base majority voting is able to answer the easier questions accurately, however it falls short for the harder questions. The success graph shows an increase on the question levels 10 and 11, but it is due to the fact that the number of questions we have for that levels in our dataset is less.

V. App based Classification of the Android Users for MCQA

In this section, we evaluate how the installed apps and the users' question answering success is correlated. In order to explore this relationship, we classify the users into 30 different user groups based on the apps installed on their devices[d]. The user groups are named after the app type categories given in Table I, and they have one-to-one correspondence with those categories. After this classification, we perform majority voting inside each user groups to measure their success.

During the classification phase, our objective is to maximize the success of the best performing user group for each question level. However, we also measure how the least successful user group performs to efficiently design our MCQA algorithms. Here we do not include the details of how each individual user group performs because of the space constraints. But still, only measuring the performances of the most and the least successful user groups gives us enough clues on how to design our MCQA algorithms.

Next, we introduce and compare four different classification techniques to classify the users based on the apps accurately. Figure 4 shows the performances of these classification techniques by question level compared to the base majority voting algorithm we defined in Section IV.

A. Basic User Classification

In basic user classification method, a user belongs to a group if she has at least one application of the corresponding app type category installed on her device. Figure 4(a) shows how the best and the worst user groups perform by question level when using this classification method.

It is clear from the graph that, although the best performing user groups slightly outperform the base majority voting, the overall success remains parallel to it.

B. Weighted User Classification

Weighted user classification method clusters the users based on the number of the apps installed from each corresponding app type category on their devices. For example, consider a device which has total of *N* number of apps installed, where *k* number of them are of the app type category *A*. Then, for the user group U_A, the user's response weight W_{U_A} would be:

$$W_{U_A} = \frac{k}{N} \tag{2}$$

Hence, if a user has more apps of a particular app type category, then her answer will have more significance in the corresponding user group. Notice that, in this classification method, each user response has different weight for a given question based on the user's response weight W_{U_A}.

Figure 4(b) shows the performances of the best and the worst user groups by question level when the weighted user classification is used.

C. Significant User Classification

We claim that the more apps installed from an app type category, the more the user is interested in that type of apps. In order to leverage this fact, we designed our significant user classification method. In this method, we define a *minimum number of apps* threshold, and classify the users based on this criteria. After some trials on our dataset, we determined this threshold as 5. Therefore, in this classification method, a user belongs to a group if she has more than 5 apps of the corresponding app type category installed on her device.

Figure 4(c) shows how the best and the worst user groups perform by question level when the significant user classification method is used. Note that, as the classification is significant when using this method, some of the user groups have less number of users. As a result, those groups do not have an answer for each and every question in our dataset. Therefore, in our analysis for significant user classification method, when a user group does not have an answer for a given question, we consider it as a non-correct answer.

Figure 4(c) reveals that, the better classification of the users increases the success of the user groups. It is also clear from the graph that, the best performing user groups are significantly better when using this classification method compared to the *basic* and the *weighted* user classification methods. Furthermore, the sharp success decrease for the higher question levels disappears when this method is used.

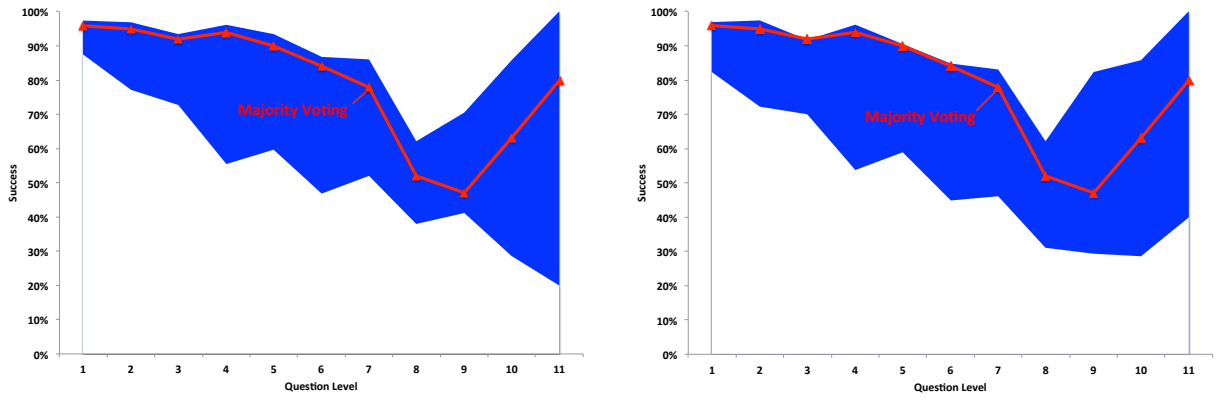
D. Competent User Classification

Competent user classification method clusters the users identical with the *significant user classification* method. Namely, in this method, a user belongs to a group if she has more than 5 apps of the corresponding app type category installed on her device. However, these two classification methods differ from each other based on how we analyze them.

Notice that, when analyzing *basic*, *weighted* and *significant* user classification methods, our objective is to answer all the questions in our dataset within each of the user groups. On the other hand, in the *competent user classification* analysis, we analyze how the user groups perform on the questions they send us answers. Therefore, in this classification method, some user groups will not be able to answer some of the questions, but they will be competent when they have an answer for a question.

Figure 4(d) shows the success of the user groups by question level when the competent user classification method is used. The graph reveals that, for each question level, there exist at least one user group which has all of its answers correct.

[d]Note that, although there are total of 34 application categories, 4 categories do not appear in our dataset as seen on Table I. Therefore we define 30 user groups instead of 34.



(a) Basic user classification

(b) Weighted user classification

(c) Significant user classification

(d) Competent user classification

Figure 4. Success of the user groups using our classification methods: Given a question level, all the user groups have a success rate inside the colored area. Namely, area boundaries for each question level are set by the success rates of the best and the worst performing user groups for that question level, based on the user classification methods.

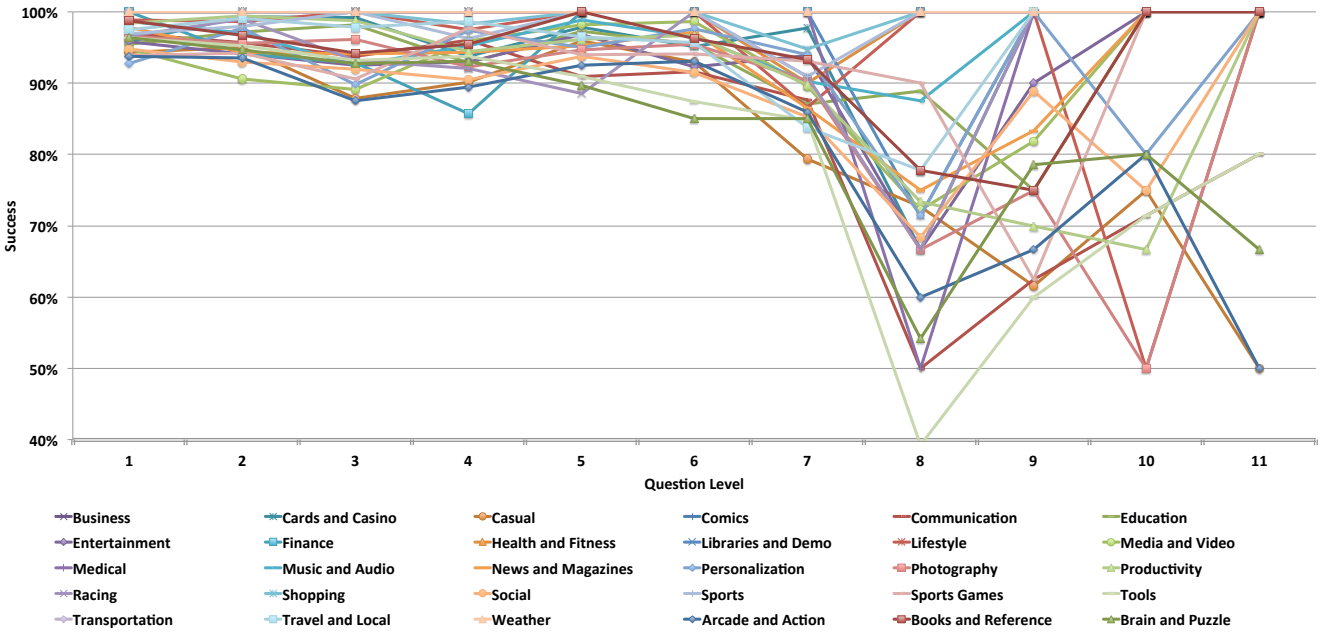
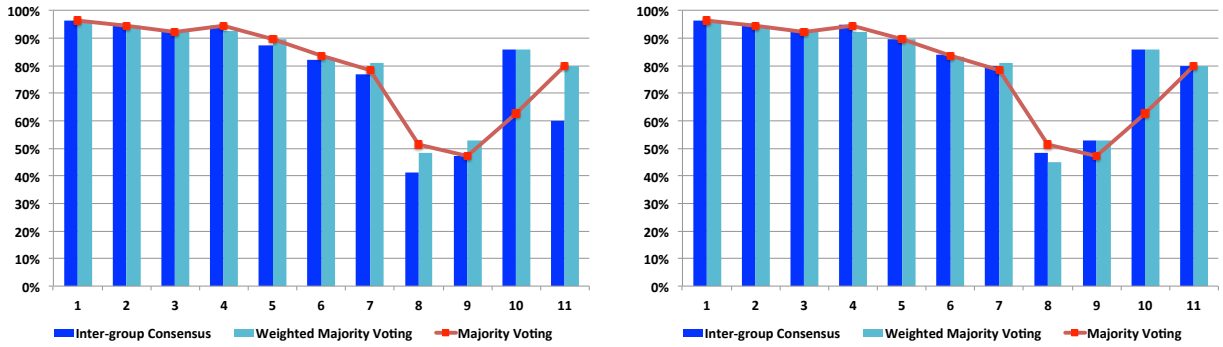


Figure 5. (This figure corresponds to Figure 4(d)) Success of the user groups in detail when using competent user classification method: This figure and thus Figure 4(d) show the success rates among only the answered questions by a user group, as opposed to the Figures 4(a), 4(b) and 4(c) which show the ratio of the correct answers to the all questions.



(a) Algorithms based on Basic user classification

(b) Algorithms based on Weighted user classification

(c) Algorithms based on Significant user classification

(d) Algorithms based on Competent user classification

Figure 6. Success of the inter-group consensus and the weighted majority voting algorithms by question level when using our user classification methods

VI. Crowd Playing "Who Wants to be a Millionaire?"

In this section, we first present two MCQA algorithms in order to investigate how our classification methods described in Section V affect the performances of the algorithms. Then using our observations, we design and introduce our *super-player algorithm*.

For each of the four classification methods, we execute the below two MCQA algorithms, and measure their success rates. The success of an algorithm is defined as the percentage of the correctly answered questions in a given question level. Figure 6 shows how these two algorithms perform based on the classification methods, and how they are compared to our base majority voting algorithm.

Algorithm 1: Inter-group Consensus

The inter-group consensus algorithm has two phases. In phase one, the algorithm classifies the users into the user groups based on the classification methods described in Section V. Then it performs majority voting inside the user groups to define an answer for each group. In phase two, the inter-group consensus algorithm performs majority voting among the user groups where each group has a single vote. The result of this voting in the second phase is the final answer selected by the algorithm.

Notice that, when using the *significant* and the *competent* user classification methods, some of the user groups do not have an answer for some of the questions in our dataset. In case a user group does not have an answer, if the algorithm is using the *significant* user classification method, then it randomly votes for a choice as a substitute during its phase two majority voting. It is due to the fact that, when using the *significant* user classification method, all the user groups have to answer all of the questions in our dataset. On the other hand, if the algorithm is using the *competent* user classification method, it ignores the non-answering user groups during the phase two majority voting, namely those non-answering groups would have no votes for the final answer.

Algorithm 2: Weighted Majority Voting

The weighted majority voting algorithm defines the user groups based on the classification methods in Section V. Then for a given question, it performs majority voting among all the users, where every user has different answer weights based on their user groups and the observed average group success rates[e].

For example, consider a user who is classified into two user groups *A* and *B*, and the observed successes of these two groups for the question level l are $A_l\%$ and $B_l\%$ respectively. Then the weight *W* of the user's answer for the question level l would be:

$$W_l = \left(\frac{A_l + B_l}{100}\right)/2 \quad (3)$$

Finally, the weighted majority voting algorithm selects the final answer for the given question by performing majority

[e]In order to incorporate these success rates into the algorithm, we used two-fold cross validation [9].

voting on those per user weighted answers.

Algorithm 3: *Our Super Player*

A. ***Design Considerations:*** *Observations that led to the super player algorithm*

- Performance of the Weighted Majority Voting Algorithm

It is clear from the Figure 6 that, regardless of which classification method it uses, the weighted majority voting algorithm performs similar to the base majority voting. It is due to the fact that the observed group success rates are not significantly distant from each other. Even taking the squares of the observed success rates does not provide enough separation, and the performance of the algorithm remains similar. This reveals that, *weighting user answers does not make much difference*.

- Performance of the Inter-group Consensus Algorithm when using the Significant User Classification Method

As seen on Figure 4, the user groups individually perform better when using *significant* user classification compared to the *basic* and the *weighted* user classification methods. On the other hand, inter-group consensus algorithm performs similar for all the three classification methods as seen on Figure 6. Namely, when using the *significant* user classification in the inter-group consensus algorithm, even though the user groups individually know the correct answer, they cannot agree on the correct answers successfully. This unveils that *the user groups are successful on dissimilar question sets*.

- Performance of the Inter-group Consensus Algorithm when using the Competent User Classification Method

As seen on Figure 6(d), the inter-group consensus algorithm when using the competent user classification method outperforms all other methods (note that the question level 11 is rare, thus it is not statistically significant). Based on the fact that the user groups are successful on dissimilar question sets when using the significant user classification, we infer that *the user groups have an answer on dissimilar question sets* when using the competent user classification (recall that the non-answering groups have no votes for the final answer when using the competent user classification).

B. ***Our Super Player Algorithm:*** *Selective User Groups*

In order to leverage the success of the inter-group consensus algorithm when using the competent user classification method, and the fact that *the user groups are successful on dissimilar question sets*, we designed the following algorithm:

Given a question, the algorithm searches for the most successful user group of the same question level. In order to calculate the user groups success map, we use a subset of our questions as our training data[f] and performed the *competent user classification* analysis that we explained in Section V-D.

After finding the most successful user group, the *selective user groups* algorithm performs majority voting inside that group. The selected choice by this majority voting is the final answer of the *selective user groups* algorithm.

[f]We used two-fold cross validation for this algorithm [9].

Data: user groups success map by question level, question
Result: answer to the question
while *has next user group* **do**
  get the next most successful user group as the current user group;
  if *the current user group performs better than the base majority voting* **then**
    if *the current user group can answer* **then**
      perform majority voting on the current user group;
      return answer;
    end
  else
    break the while loop;
  end
end
perform the base majority voting;
return answer;

Algorithm 3: MCQA using Selective User Groups

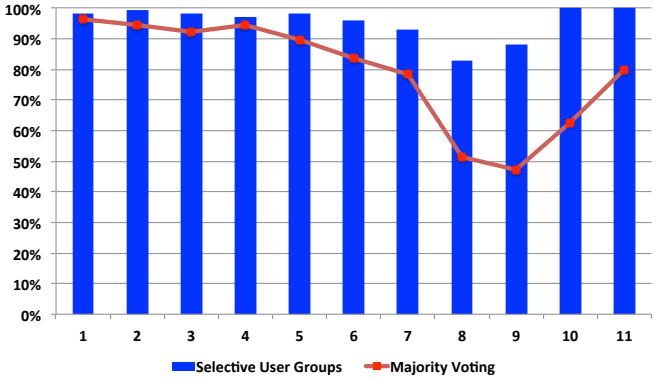


Figure 7. MCQA using the Selective User Groups algorithm success by question level

Note that, when using the competent user classification method, some user groups will not be able to answer some of the questions. Therefore, if the most successful group does not have an answer for the given question, then the *selective user groups* algorithm looks for the second best performing group for that question level to answer the question. And if that group also cannot answer the question, the algorithm keeps searching for the next group until no further group remains that perform better than the base majority voting. In the worst case, the *selective user groups* algorithm answers the question using the base majority voting. Algorithm 3 shows this execution flow.

Figure 7 shows how the *selective user groups* algorithm performs by question level. As it is clear from the graph that this algorithm performs significantly better than the base majority voting especially for the higher level questions where the base majority voting falls short. Note that the *selective user groups* algorithm also outperforms our second best algorithm

which is *the inter-group consensus algorithm when using the competent user classification method* shown in Figure 6(d). On the other hand, *the inter-group consensus algorithm when using the competent user classification method* is remarkable in the sense that it does not require any training data.

VII. Related Work

In recent years, ML and IR communities have been producing large and growing literature on question answering. Today, there are many studies on question answering in the form of game playing. Some of these systems rely on the AI based reasoning and NLP. IBM Watson's [3] Jeopardy challenge is the best known example of such systems. In order to outperform the human contestants, IBM Watson's Jeopardy system leverages NLP, ML and IR techniques.

MCQA is also drawing researchers' attention and there are several recent works on MCQA. Lam et al. [10] describe how search engines can be utilized to play WWTBAM game. Their basic method is to race the hit numbers of *question+choice* pairs by searching the possible combinations of such pairs on the search engines. With this naive strategy, out of 635 questions, their system is able to answer 44.4% of them using *MSN.com* and 55.6% of them using *Google*. In their advanced approach, they use the proximity of the question and the choice keywords as they appear in the search results. With the help of this strategy, their system is able to answer 68.9% of the questions. In [11], authors work on Japanese WWTBAM questions, and they try to detect the question phrase to decouple the choices before performing the hit race. With this keyword association, their system is able to answer the questions with 79% overall success rate. However, in another work [12], the same method performs 45% on Arabic WWTBAM dataset. The authors of the latter work, on the other hand, are able to pull the performance up to 55% with an improved method on Arabic questions. Their system also performs 62% on English questions. In a recent work [13] on WWTBAM, authors build a virtual player to play the game using Italian questions. Similar to IBM Watson's Jeopardy system, their virtual player leverages the information retrieved from Wikipedia. The overall success rate of their system is 76.33%. However, in none of the works above, the performance is proportional to the difficulty of the questions, because they are based on IR techniques as opposed to our system that leverages the crowd's intelligence.

Crowdsourcing has been employed for answering subjective, relative, or multidimensional location-based queries for which the traditional search engines perform poorly [14], [15]. Similar to our previous location-based question answering work [14], there are several other question answering systems. AskMSR [16] leans on the redundant data on the web instead of the complex NLP techniques. ChaCha [17] and SMS-Find [18] are other examples of SMS-based question answering systems. ChaCha tasks the users based on their location and the user statistics. SMSFind leverages the search engines to answer the questions automatically. It tries to find the best matching answer by using both IR and NLP techniques. Although SMSFind is not a crowdsourced answering system, it is remarkable in the sense that the authors point out what type of questions are ambiguous to their system and where the human intelligence is needed. In another work, CrowdDB [19], authors present an SQL-like query processing system to collect large amount of data with the help of microtask-based crowdsourcing. CrowdDB is piggybacked to Amazon Mechanical Turk [20].

VIII. Future Work

Our work reveals that the app based user classification improves the success of our MCQA algorithms. Recall that, it is due to the fact that *the user groups are successful on dissimilar question sets*. Therefore, we foresee that the classification of the questions will further improve our performance. As a future work, we plan to categorize our question set and then investigate how this question categorization changes the success of our algorithms.

Furthermore, in the scope of this paper, we classify the users into the user groups where many users belong to multiple groups. We plan to employ some ML clustering algorithms to classify our user base where each user belongs to a single cluster. Using the question categorization and this clustering together, we plan to extend our understanding of the targeted MCQA dynamics more.

References

[1] "Google now," http://www.google.com/now.
[2] "Apple siri," http://www.apple.com/ios/siri.
[3] D. Ferrucci, E. Brown, J. Chu-Carroll, J. Fan, D. Gondek, A. A. Kalyanpur, A. Lally, J. W. Murdock, E. Nyberg, J. Prager *et al.*, "Building watson: An overview of the deepqa project," *AI magazine*, vol. 31, no. 3, pp. 59–79, 2010.
[4] C. H. Lin, Mausam, and D. S. Weld, "Crowdsourcing control: Moving beyond multiple choice," in *UAI*, 2012, pp. 491–500.
[5] "Who wants to be a millionaire?" http://en.wikipedia.org/wiki/Who_Wants_to_Be_a_Millionaire?
[6] "Google play page of application," https://play.google.com/store/apps/details?id=edu.buffalo.cse.ubicomp.crowdonline.
[7] B. I. Aydin, Y. S. Yilmaz, M. F. Bulut, and M. Demirbas, "Crowdreply: A crowdsourced multiple choice question answering system," *ACM/IEEE IPSN '13 3rd International Workshop on Mobile Sensing: The Future, brought to you by Big Sensor Data*, 2013.
[8] "Google play store," https://play.google.com/store/apps.
[9] F. Mosteller and J. W. Tukey, *Data analysis, including statistics*, 1968.
[10] S. K. Lam, D. M. Pennock, D. Cosley, and S. Lawrence, "1 billion pages= 1 million dollars? mining the web to play" who wants to be a millionaire?"," in *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2002, pp. 337–345.
[11] E. Sumita, F. Sugaya, and S. Yamamoto, "Measuring non-native speakers' proficiency of english by using a test with automatically-generated fill-in-the-blank questions," in *Proceedings of the Second Workshop on Building Educational Applications Using NLP*, ser. EdAppsNLP 05. Stroudsburg, PA, USA: Association for Computational Linguistics, 2005, pp. 61–68.
[12] R. Awadallah and A. Rauber, "Web-based multiple choice question answering for english and arabic questions," in *Advances in Information Retrieval*. Springer, 2006, pp. 515–518.
[13] P. Molino, P. Basile, C. Santoro, P. Lops, M. de Gemmis, and G. Semeraro, "A virtual player for who wants to be a millionaire? based on question answering," in *AI* IA 2013: Advances in Artificial Intelligence*. Springer, 2013, pp. 205–216.
[14] M. F. Bulut, Y. S. Yilmaz, and M. Demirbas, "Crowdsourcing location-based queries," in *PerCom Workshops*, 2011, pp. 513–518.
[15] M. Demirbas, M. A. Bayir, C. G. Akcora, Y. S. Yilmaz, and H. Ferhatosmanoglu, "Crowd-sourced sensing and collaboration using twitter," in *WOWMOM*, 2010, pp. 1–9.
[16] E. Brill, S. Dumais, and M. Banko, "An analysis of the askmsr question-answering system," in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, ser. EMNLP '02. Stroudsburg, PA, USA: Association for Computational Linguistics, 2002, pp. 257–264.
[17] http://www.chacha.com.
[18] J. Chen, L. Subramanian, and E. Brewer, "Sms-based web search for low-end mobile devices," in *Proceedings of the sixteenth annual international conference on Mobile computing and networking*, ser. MobiCom '10. New York, NY, USA: ACM, 2010, pp. 125–136.
[19] M. J. Franklin, D. Kossmann, T. Kraska, S. Ramesh, and R. Xin, "Crowddb: answering queries with crowdsourcing," in *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data*, ser. SIGMOD '11. New York, NY, USA: ACM, 2011, pp. 61–72.
[20] "Amazon mechanical turk: Artificial artificial intelligence," https://www.mturk.com/mturk.