

Crowdsourced Query Processing on Microblogs

Weikeng Chen¹(✉), Zhou Zhao², Xinyu Wang¹, and Wilfred Ng¹

¹ The Hong Kong University of Science and Technology, Hong Kong, China

{wchenad,xwangau,wilfred}@cse.ust.hk

² Zhejiang University, Hangzhou, China

zhaozhou@zju.edu.cn

Abstract. Currently, crowdsourced query processing is done on reward-driven platforms such as *Amazon Mechanical Turk (AMT)* and *Crowd-Flower*. However, due to budget constraints for conducting a crowdsourcing task in practice, the scalability is inherently poor. In this paper, we exploit microblogs for supporting crowdsourced query processing. We leverage the social computation power and decentralize the evaluation of the crowdsourcing platforms queries towards social networks. We propose a new problem of minimizing the cost of processing crowdsourced queries on microblogs, given a specified accuracy threshold of users' votes. This problem is NP-hard and its computation is #P-hard. To tackle this problem, we develop a greedy algorithm with a quality guarantee. We demonstrate the performance on real datasets.

1 Introduction

Crowdsourcing techniques [10, 15, 27–36] have attracted considerable attention due to their effectiveness in many applications such as entity resolution and image detection. An essential property of crowdsourcing is that the technique relies on a human workforce to at least partially complete an evaluation of the queries. Typically, a crowdsourcing application publishes its queries assigning with a fixed reward to the workers on *Amazon Mechanical Turk*¹ and *Crowd-Flower*². Each crowdsourced query is then assigned with a fixed reward to the workers.

However, humans are prone to error and may provide poor quality crowdsourcing results. To address the problem, crowdsourcing applications often enroll a number of workers to process the replicated queries. If the collected results are conflicting, the majority vote is adopted to determine which is correct. However, the replication strategy may not be able to fully handle the diversity of answers. Suppose the tasks involved are hard, we have to enroll more workers to reduce the diversity of answers. The cost of this *Human Intelligence Task (HIT)* could then be very high. Thus, a limitation of the existing crowdsourcing approach is, that we may not have a sufficient number of workers to process the query under the budget constraint, which results in poor answer quality.

¹ Amazon Mechanical Turk (or simply AMT) platform at <https://www.mturk.com>.

² CrowdFlower platform at <https://www.crowdflower.com>.

To tackle this problem, we formulate a new problem of processing crowdsourced queries on microblog, which provides new incentives to encourage microblog users to process crowdsourced queries. By developing a new crowdsourcing model, we aim to reduce both the diversity of the answers and the cost of processing. We focus on addressing the following main issues:

- **Answer Diversity.** If the number of replicated queries are too few, we may not have enough confidence to infer a reliable answer. On the other hand, if we replicate too many queries, we may have to suffer high cost.
- **Incentive Mechanisms.** Crowdsourcing workers may be reluctant to process any queries until they know they will receive a reward. Thus, we aim to design an incentive mechanism that is able to reduce the workforce cost and at the same time, meet a given specified accuracy threshold.
- **Query Sharing.** We utilize *the word of mouth* effect to model the worker behavior of query sharing on microblogs. Intuitively, microblog users are more willing to answer a crowdsourced query under a social influence and to send messages to an interested group. Thus, crowdsourced queries can be diffused more efficiently and effectively over microblogs.

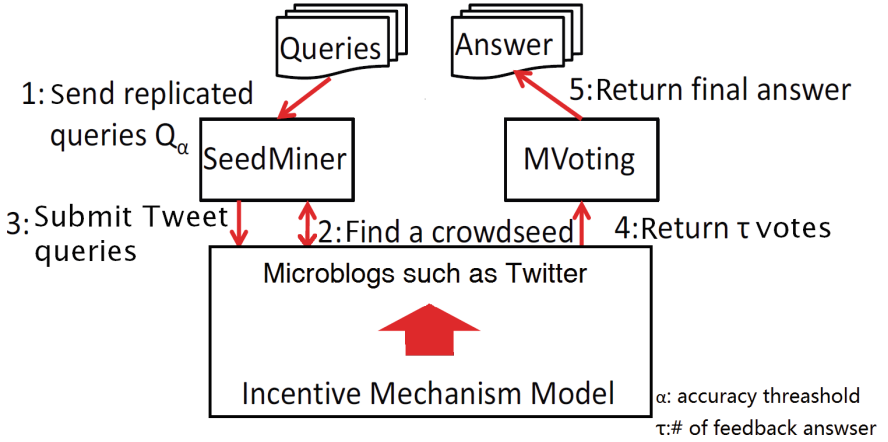


Fig. 1. A microblog-based crowdsourcing system

There are already some works studying the problem of answer diversity [12, 18, 20]. However, they all rely on centralized platforms such as *AMT* or *CrowdFlower*. In contrast, we study how to address the problem by exploiting the social influence of microblogs.

The general process of crowdsourced query processing on microblogs is given in Fig. 1. For instance take the crowdsourced query Q “Is Paris the City of Lights?” and the specified *accuracy threshold* α . Owing to the conflicting answers, we should have a sufficient number of replicated queries Q such that the accuracy of the final answer obtained from using the *majority voting rules* (i.e. the MVoting module) is greater than α . Let Q_α be the set of such replicated

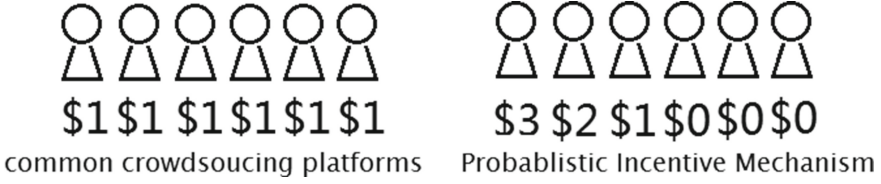


Fig. 2. The difference between common crowdsourcing platforms and the Probabilistic Incentive Mechanism Model

queries. The estimation of the number of queries in Q_α is based on the historic degree of the skill of the microblog users (Fig. 2).

In the first step, we take Q_α as the input to *SeedMiner*. The *SeedMiner* estimates the minimum crowdseed size needed to diffuse this query in microblogs such that we can have τ feedback. The *SeedMiner* algorithm is based on the “word of mouth” effect [13] of query diffusion on microblogs. In the second step, the *SeedMiner* returns the crowdseed and we issue the query to the crowdseed in the third step. We collect answers incrementally from microblogs in the fourth step. Finally, we fuse the feedback and deliver the correct solution online using *MVoting*. In the system, we develop a lottery based incentive mechanism to encourage users to answer and “retweet” the crowdsourced query. We employ the build-in lottery function in the third party [22] to our system.

Contributions. We tackle the problem of crowdsourced query processing on microblogs. Specifically, we make the following contributions.

- We formulate the problem of crowdsourced query processing on microblogs and exploit the query diffusion process that shifts towards decentralized crowdsourcing platforms.
- We design a new lottery based incentive mechanism to encourage users to answer and “retweet” the crowdsourced query on microblogs.
- We propose a sampling-based greedy algorithm that tackles the problem with a quality guarantee.
- We demonstrate the performance of our new approach to crowdsource query processing based on its real application on *Epinion*, *NetHEPT* and *Twitter*.

This paper is organized as follows. Section 2 introduces the crowdsourcing models. Section 3 formulates the problem of crowdsourced query processing on microblogs and analyze the complexity of the problem. Section 4 then surveys the related work while Sect. 5 presents the details of our algorithm. Section 6 presents the experimental results and we conclude the paper in Sect. 7.

2 Crowdsourcing Model

In this section, we first introduce the popular voting model, which is widely used in a crowdsourcing environment. Then we propose the incentive model and the diffusion model in our approach.

Voting Model. Given a query, the judgement among the crowd may be different. As a result, the human answers for the query posed on microblogs may easily conflict. To resolve this problem, we develop a voting model that consists of a *majority voting rule* given by

$$f(V) = \begin{cases} 1 & \text{if } \sum_{v_i \in V} v_i \geq \frac{\tau+1}{2} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

where the vote v_i is the answer of user u_i and V represents a collection of τ votes (or equivalently, τ feedback answers). Among the conflicting answers from the votes, we choose one that is supported by more than half of the votes. For ease of presentation, we consider the crowdsourced decision-making query in this paper. The vote v_i can be a binary random variable (i.e. either 0 or 1). It can also easily be extended to queries with K possible answers. In such a scenario, we can also assume the answer with more than half the votes is the correct one.

However, the output of the *majority voting rule* may not be reliable, if the number of votes is too few. On the other hand, the cost is high if we enroll too many votes. We thus propose a probabilistic model to estimate the number of replicated queries that are sufficient to make the *majority voting rule* reliable.

Suppose the accuracy of users' votes on microblogs that have processed the query are $\{a(u_1), \dots, a(u_\tau)\}$, where $a(u_i)$ is the vote accuracy of user u_i (i.e. the probability of vote v_i being correct). We utilize the historic records of the degree of skill of the users to estimate the voting accuracy. In this paper, we do not focus on the estimation of the degree of skill of the users, as a variety of solutions have already been proposed in the literature [15, 20, 28, 36].

Given a collection of votes V of size τ , we consider the correctness probability of the *majority voting rule* as $p(f(V))$. We denote the random variable s_A consisting of all the accurate votes in V (i.e. $s_A \subseteq V$). The output result of the majority voting is correct only when at least half of the votes are accurate (i.e. $|s_A| \geq \frac{\tau+1}{2}$). Thus, the correctness probability of the *majority voting rule* is given by

$$\begin{aligned} p(f(V)) &= Pr(|s_A| \geq \frac{\tau+1}{2}) = \sum_{k=\frac{\tau+1}{2}}^{\tau} Pr(|s_A| = k) \\ &= \sum_{k=\frac{\tau+1}{2}}^{\tau} \sum_{s_A \in F_k} \prod_{u_i \in s_A} a(u_i) \prod_{u_j \notin s_A} (1 - a(u_j)) \end{aligned} \quad (2)$$

where F_k comprises all possible combinations of users giving accurate votes for s_A with size k . We note that $1 - p(f(V))$ is a cumulative Poisson binomial distribution, since the accurate probability of each vote v_i is different.

We consider the accuracy of the result by the *majority voting rule* as the expectation of correctness probability (i.e. $E[p(f(V))]$). Then, the expected correctness of the *majority voting rule* is given by

$$E[p(f(V))] = \sum_{k=\frac{\tau+1}{2}}^{\tau} \binom{\tau}{k} \mu^k (1-\mu)^{\tau-k} \quad (3)$$

where μ denotes the average degree of skill of the users (i.e. the average accuracy of the votes). By using the Chernoff Bound, the lower bound of the expected correctness is given by

$$\sum_{k=\frac{\tau+1}{2}}^{\tau} \binom{\tau}{k} \mu^k (1-\mu)^{\tau-k} \geq 1 - e^{-\frac{2\tau(\mu-\frac{1}{2})^2}{4\mu}} \quad (4)$$

Then, we let $1 - e^{-\frac{2\tau(\mu-\frac{1}{2})^2}{4\mu}} \geq \alpha$, where α is the specified accuracy and the number of replicated queries is given by

$$\tau \geq \frac{-4\mu \ln(1-\alpha)}{2(\mu - \frac{1}{2})^2} \quad (5)$$

Thus, the next goal is to seek τ users to answer the crowdsourced query. In the next section, we present incentive mechanisms to tackle this problem.

Incentive Mechanism Model. To encourage users to answer an imposed crowdsourced query, the existing centralized crowdsourcing platforms commonly pay each user a fixed amount of money as a reward for completing the task.

However, we find that such fixed reward incentive mechanisms do not work well on microblogs. In our experiment, we firstly distribute several crowdsourced queries with a reward of \$0.05 to the crowd on microblogs. We found that few users answer these queries. Our observations are as follows: unlike the workers from *AMT*, users on social networks are reluctant to answer any unexpected problem with too little reward (e.g. \$0.05). On the other hand, microblog users do not primarily aim to earn money but rather gain social interactions.

In this paper, we devise and evaluate a new incentive mechanism for crowdsourced query processing based on the platform *Sojump* [22]. *Sojump* is a well established Q&A platform which is able to process a reward payment. Similar to other common Q&A systems, we generate problem sheets with a URL under the homepage of *Sojump* and specify the payment conditions. In our model, the payment condition is that a user completes the problem sheet and shares the URL of the sheet in the microblog.

Given a crowdsourced query budget B , we devise a *probabilistic incentive mechanism* in order to encourage the crowd on the microblog to answer and “retweet” the problem sheet. For example, we specify the budget of 6 crowdsourced queries to \$6 in this experiment. First, we decompose the total budget into three rewards such as \$3, \$2 and \$1, as illustrated in Fig. 1. Suppose that the number of replicated crowdsourced queries is τ , we set the probability of getting each reward to $\frac{1}{\tau}$.

Crowdseed. The basic idea for processing crowdsourced queries under our incentive mechanism model is to issue the queries to a large number of users on microblogs. However, this approach incurs two problems. First, the manager of the microblog may suspect our application to be a type of spammer and forbid its usage. Second, the users on microblogs may become annoyed when they receive unexpected problem sheets from unknown sources.

To tackle the above problems, we include the concept of *crowdseed* in the process such that, after we issue the problem sheets to the users in the crowdseed, the problem sheets may be diffused through their microblog relations to τ other users within the probabilistic incentive mechanism model. Our system encourages users to subscribe to crowdseed where the users can set the expected reward to accept the posting of crowdsourced queries. We consider the subscription cost of user u_i as $c(u_i)$. Then, the initialization cost of issuing the crowdsourced query to the microblog is given by $c(S) = \sum_{u_i \in S} c(u_i)$, where S represents the *crowdseed* (i.e. a set of users who are willing to accept our crowdsourced queries as the seed). The seed cost $c(S)$ depends on the specified accuracy of the crowdsourced query (i.e. α).

Diffusion Model. Under the word of mouth effect [14], users' behaviors are probabilistically influenced by their friends. For example, microblog users may follow their friends' actions by "retweeting" the messages being shared. Due to this observation, we propose a probabilistic model based on the well-known *word of mouth* effect to model the crowdsourced query diffusion.

We denote the event that user u_i successfully diffuses the crowdsourced query to his/her friend u_j as $I(u_i, u_j)$. In practice, we found that microblog users tend to share "tweets" from their close friends. Based on this observation, we explore the query diffusion probability model between two users based on their closeness in microblogs. In this work, we utilize the *Jaccard Distance* of two users' friends to measure their closeness. The closeness of two users u_i and u_j is given by

$$J(u_i, u_j) = \frac{|N(u_i) \cap N(u_j)|}{|N(u_i) \cup N(u_j)|} \quad (6)$$

where $N(u_i)$ denotes the set of users that u_i follows and $|N(u_i) \cap N(u_j)|$ is the number of common following users of u_i and u_j .

We propose to use the popular *Sigmoid* function to explore the relationship between query diffusion and the closeness of users. We denote the probability of query diffusion between two users as $p(I(u_i, u_j))$. Then, the formula of the crowdsourced query diffusion model from user u_i to user u_j is given by

$$p(I(u_i, u_j)) = \frac{1}{1 + e^{aJ(u_i, u_j) + b}} \quad (a < 0, b > 0) \quad (7)$$

where a and b are the parameters of the probabilistic query diffusion model.

To estimate the values of parameters a and b , we conduct user study by posing 700 queries on *Twitter*. We collect the data in the format of

$(u_i, u_j, N(u_i), N(u_j), I)$ where I is the indication of the query diffusion. Next, we aggregate the collected data in the format of (J, p) where J is the *Jaccard Distance* value and p is the diffusion probability. We notice that Eq. 7 implies that $\ln(\frac{1}{p} - 1) = aJ + b$. Next, we employ the least square method to estimate the parameters a and b , using a transformed set of pairs such as $\{\ln(\frac{1}{p} - 1), J\}$.

Using Eq. 7, we transform the microblog into a probabilistic graph given by $G = (V, E, P)$, where V consists of all users, E is composed of all relations and P records the pairwise diffusion probability of two users (i.e. $p(I(u_i, u_j))$).

We consider the event in which the crowdsourced query diffuses from the crowdseed S to any user u_k as a graph reachability problem. However, the query diffusion between two users is uncertain. For a probabilistic graph with E edges, we have 2^E possible cases of query diffusion which are denoted as $\{G_1 = (V, E_1), \dots, G_{2^{|E|}} = (V, E_{2^{|E|}})\}$. Then, the query diffusion from the crowdseed S to any user u_k is given by

$$p(I(u_k)|S) = \sum_{i=1}^{2^{|E|}} p(G_i) R_{G_i}(u_k, S) \quad (8)$$

where $p(G_i)$ is the probability of the i th query diffusion case. The value of $p(G_i)$ can be computed by the product of the edge probabilities. We denote $R_{G_i}(u_k, S)$ as the indication of the reachability from the crowdseed S to the user u_k in case G_i (i.e. either zero or one). In other words, the query diffusion probability is the sum of the probability of cases that the query from crowdseed S can diffuse to the user u_k . We consider the expected diffusion size of the crowdseed S in the probabilistic graph G as $\delta(G|S)$. Then, the expected size $\delta(G|S)$ is given by

$$\delta(G|S) = \sum_{i=1}^{|V|} p(I(u_i)|S) \quad (9)$$

where $|V|$ denotes the number of users in G .

3 Problem Statement

We formulate the problem of crowdsourced query processing on a microblog as *Crowdseed Selection* as follows. In a nutshell, we aim to seek a crowdseed set S from the microblogs such that: (1) we could have at least a collection of τ feedback answers, and (2) the cost of the crowdsourced query Q can be reduced as much as possible.

Problem 1 (Crowdseed Selection). Given a probabilistic graph $G(V, E, P)$ of the microblog and a crowdsourced query Q , we aim to find a crowdseed set such that Q can be processed with an expected accuracy larger than α .

However, Theorems 1 and 2 show that the complexity of this problem is NP-hard and even the computation complexity of the expected query diffusion of a crowdseed set S is #P-hard. The detailed proof can be found in Appendix [1].

Theorem 1. *The problem of Crowdseed Selection is NP-hard.*

Theorem 2. *The computation complexity of expected query diffusion of a crowdseed set S is #P-hard.*

4 Related Work

In this section, we survey some proposed crowdsourcing systems and introduce some work about processing various kinds of crowdsourcing queries.

Crowdsourcing Systems. Many crowdsourcing database systems like *Qurk* [17], *Deco* [19] and *Hog* [4] have recently been proposed as the plug-in components for traditional database systems. These systems integrate existing crowdsourcing platforms such as *Amazon MTurk* and *CrowdFlower* as an external data source. The crowdsearcher [2], a novel search paradigm, embodies crowds as first-class sources for the information seeking process. The Crowdturfing system [25] aims to study and understand the Crowdturfing campaigns in today’s Internet. The work ZenCrowd [8] systematizes and automatizes manual matching techniques by dynamically creating micro matching tasks and by publishing them on a popular crowdsourcing platform. The work Cogos [9] leverages Twitter Lists to find topic experts in Twitter.

Queries with the Crowd. The unreliability of workers is a significant challenge for query processing using the crowdsourcing strategy, thus, different approaches have been proposed to tackle the problem of conflicting answers. The work in [6, 12, 24] resolves the ordinal query problem of conflicting rankings. The work in [3, 16, 18] studies the screen query problem of conflicting answers. The work in [23] studies the crowdsourced enumeration query. The work in [21] studies the query-driven schema expansion. The work in [26] studies the crowdsourced join query to find all pairs of matching objects from two collections. The work in [7] studies the top-k and group-by queries with the crowd. The work in [13] studies how to choose the right question to answer the planning query with the crowd.

Some recent works [3, 15] have studied the crowd selection problem on microblog and built a probabilistic voting model to estimate the reliability of the enrolled crowd. The dynamic programming based and greedy based algorithms are proposed to select the crowd. The selection of the crowd is only based on the reliability of the workers.

However, none of the above-mentioned works utilize the power of social influence for processing crowdsourced queries on microblogs. Our work shows that Twitter users send crowdsourced queries to their friends and that people also answer the Twitter queries based on their friendships. By taking the social influence into consideration, we further study how to mine crowdseed for crowdsourced query processing on microblogs.

5 Algorithms

In this section, we show how to tackle the complexity of the problem of *Crowdseed Selection*. We first define an objective function based on the selected crowdseed set S and then propose a greedy algorithm in order to maximize the function.

We aim to select a crowdseed set S such that τ feedback answers are obtained from the users. Thus, we set the constraint of the crowdseed set S at $\delta(G|S) \geq \tau$ and formulate the objective function as: $\min_S \sum_{u_i \in S} c(u_i)$ such that $\delta(G|S) \geq \tau$, where τ is the expected query diffusion size. The value of τ can be computed by Eq. 5.

We now present a greedy algorithm that iteratively selects the subscribed users to the crowdseed set S in order to satisfy the constraint. We denote the ratio of the expected query diffusion (i.e. $\delta(G|S)$) to the cost of S (i.e. $c(S)$) as Δ_S (i.e. $\Delta_S = \frac{\delta(G|S)}{c(S)}$). In each iteration, we aim to select a subscribed user u_i to the crowdseed S that is able to maximize this ratio, given by

$$u_i = \operatorname{argmax}_{u_i} \left(\frac{\delta(G|S \cup \{u_i\})}{c(S \cup \{u_i\})} - \frac{\delta(G|S)}{c(S)} \right) = \operatorname{argmax}_{u_i} (\Delta_{S \cup \{u_i\}} - \Delta_S) \quad (10)$$

The greedy algorithm terminates until the expected query diffusion is larger than τ (i.e. $\delta(G|S) \geq \tau$).

However, given a crowdseed set S , the computation of its expected query diffusion is #P-hard. Thus, we propose a sampling algorithm to estimate the expected query diffusion efficiently and effectively. As $G = (V, E, P)$ is a probabilistic graph, the samples can be obtained by flipping the edges according to the probabilities in P . For example, we can have k sample graphs, $G_1 = (V, E_1), \dots, G_k = (V, E_k)$. Thus, the expected query diffusion can be obtained by taking the average of these sample graphs and is given by

$$\overline{\delta(G|S)} = \frac{\sum_{i=1}^k \delta(G_i|S)}{k} \quad (11)$$

where the sample G_i is a deterministic graph. The $\delta_B(G_i|S)$ is the size of the connected nodes from the set S which can be computed using the *BFS* algorithm.

We also show that the sampling algorithm can achieve (ϵ, η) approximation of estimating the expected query diffusion (i.e. $\delta(G|S)$). Using *Hoeffding's Inequality*, we have

$$Pr(|\overline{\delta(G|S)} - \delta(G|S)| \geq \epsilon) \leq 2 \exp\left(-\frac{2\epsilon^2 k^2}{\sum_{i=1}^k (|V| - 1)^2}\right) \leq \eta \quad (12)$$

where ϵ is the error rate and η is the confidence of the estimation. Then it follows that we achieve (ϵ, η) approximation if the number of samples

$$k \geq \frac{(|V| - 1)^2 \ln \frac{2}{\eta}}{2\epsilon^2 |S|^2} \quad (13)$$

where $|S|$ is the size of the crowdseed S . Details are shown in Appendix [1]. The details of the greedy and sampling algorithms are presented in Algorithms 1 and 2. First, in Algorithm 1 we transform the obtained microblog into a probabilistic graph $G = (V, E, P)$ based on the similarity between two users on Line 1. The diffusion process of the crowdsourced query is based on G . Next, we compute the expected accuracy (i.e. μ) based on the average value of the historical accuracy record of the users on Line 2. Then, we estimate the lower bound of the number of replicated queries by Eq. 5 on Line 3. When the crowdseed S is empty, the expected query diffusion is assumed to be zero (i.e. $\delta(S|\phi) = 0$). At each iteration, we select a subscribed user u_i that is able to maximize the objective function on Line 6 and add it to the crowdseed S on Line 7. Algorithm 1 terminates when the expected query diffusion of the current crowdseed (i.e. $\delta(G|S)$) is larger than the number of required replicated queries (i.e. $\delta(G|S) \geq \tau$) on Line 5.

Algorithm 1. *Crowdseed(Q_α)*

Input: Q_α : a crowdsourced query with specified accuracy α ; $a(u_1, \dots, a(u_{|V|}))$: a historic accuracy record

Output: S : a crowdseed

- 1: Build a probabilistic graph $G = (V, E, P)$ by Equation 7
 - 2: Expected accuracy $\mu \leftarrow \frac{\sum_{i=1}^{|V|} a(u_i)}{|V|}$
 - 3: Set # of replicated queries τ by Equation 5
 - 4: Set crowdseed $S \leftarrow \emptyset$
 - 5: **while** $\delta(G|S) < \tau$ **do**
 - 6: $u_i = \operatorname{argmax}_{u_i} (\Delta_{S \cup \{u_i\}} - \Delta_S)$
 - 7: $S \leftarrow S \cup \{u_i\}$
 - 8: **end while**
 - 9: **return** S
-

We illustrate the sampling process of computing the expected query diffusion $\delta(G|S)$ in Algorithm 2. In order to achieve (ϵ, η) approximation of the expected query diffusion, we choose K samples for estimation by Eq. 5 on Line 1. At each time, we sample K certain graphs from the probabilistic graph $G = (V, E, P)$ and take the average of the query diffusion of the current crowdseed S on the

Algorithm 2. *GetDiffusion(G, S)*

Input: G : a probabilistic graph; S : a candidate crowdseed

Output: $\delta(G|S)$: an expected query diffusion

- 1: Set # of samples K by Equation 13
 - 2: Set expected query diffusion $\delta(G|S) \leftarrow 0$
 - 3: **for** $i = 1 \rightarrow K$ **do**
 - 4: Sample G_i form $G = V, E, P$
 - 5: $\delta(G_i|S) \leftarrow \text{BFS}(G_i, S)$
 - 6: $\delta(G|S) \leftarrow \delta(G|S) + \delta(G_i|S)/K$
 - 7: **end for**
 - 8: **return** $\delta(G|S)$
-

samples from Lines 3 to 8. We employ the *BFS* algorithm to compute the query diffusion on certain graphs on Line 5. In the implementation, we find that the time duration of generating K sample graphs is very lengthy. To tackle this problem, we sample K certain graphs offline and store them in the memory. Thus, Algorithm 2 is able to compute query diffusion on these stored graphs without generating new samples.

6 Experimental Studies

The studies are performed in a Linux box with an 8-core Intel(R) Xeon(R) CPU X5450 3.00 GHz and 16 GB memory.

We investigate the robustness of our algorithms by varying the size of the crowdseed and the query cost.

Datasets. In order to generate the synthetic datasets, we simulate the crowd-sourced query diffusion process using three real datasets. The *Epinion* is a Who-trusts-whom network where the vertices represent the sites and the edges represent the trust relations between two sites. The *NetHEPT* is a large academic collaboration network where the vertices represent authors and the edges represent coauthorship relations. *Twitter* is microblog dataset where the vertices represent the users and the edges represent the following relationships. Thus, we build the probabilistic graph $G = (V, E, P)$ based on these three datasets in order to model the query diffusion. The statistics of these three real social graphs are given in Table 1.

Table 1. Statistics of datasets

Dataset	Epin	NetHEPT	Twitter
No. of nodes	75888	15233	11555
No. of edges	508837	58891	500000

Baseline Algorithms. We evaluate the effectiveness and robustness of the proposed algorithms using the three real social graphs aforementioned. We also propose four baseline algorithms such as *Degree-based* algorithm, *Centrality-based* algorithm, the CELF++ [11] Algorithm and the LDAG [5] The *Degree-based* and *Centrality-based* algorithms rank all the vertices based on vertices degree and centrality first. The CELF++ Algorithm and LDAG algorithm picks the most influential seeds by their definition. Then, these four algorithms select the vertices and add them to the crowdseed S according to their order.

Measurement. We demonstrate the effectiveness of the proposed algorithms by generating crowdsourced queries for each dataset with different error rate ϵ (i.e. $1 - \alpha$). We propose two quality measurements: (1) size of Crowdseed (# of seeds) and (2) cost of crowdsourced query. In the first phase, we assume that

the cost of each user joining the crowdseed S is the same. Thus, the algorithm which outputs the smallest crowdseed works the best. In the second phase, we generate the cost of each user to join crowdseed S from a uniform distribution (i.e. $Uni(1, 100)$). Then, we compare the cost of the crowdsourced query for different algorithms, where the one with the least cost is the most effective.

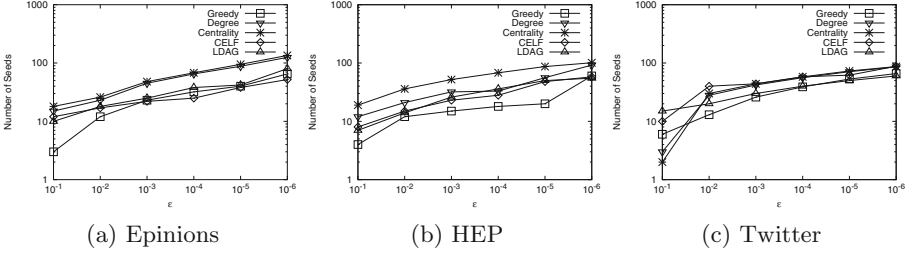


Fig. 3. Crowdseed size v.s. error rate

Crowdseed Size. Figure 3(a), (b) and (c) illustrate the size of the crowdseed of the proposed algorithms by generating six types of crowdsourced queries for each dataset with different error rate ϵ (i.e. $10^{-1}, 10^{-2}, \dots, 10^{-6}$). For example, the size two crowdseed means that we need to issue the crowdsourced query to two users as the seeds for later diffusion.

To mitigate the unreliability of the *majority voting rule*, we need to enroll many users to improve its accuracy. In the microblog, the expected query diffusion is proportional to the size of the crowdseed. Thus, as the error rate decreases, the size of the crowdseed increases. However, the size of crowdseed by the greedy algorithm is smaller than other algorithms.

The baseline algorithms select the seeds based on their independent query diffusion. However, selecting a vertex with a high diffusion value may not always increase the query diffusion of the crowdseed by much. For example, selecting a vertex with a lot of neighbors in the crowdseed may not increase the total query diffusion by much. Thus, our proposed algorithm considers the joint expected query diffusion of the selected seeds such that this problem can be avoided.

Query Cost. We study the cost of processing a crowdsourced query using our algorithms. Figure 4(a), (b) and (c) show the query cost of our proposed algorithms on different error rates. To make a fair comparison with the baseline algorithms, we penalize the users with high costs in both *Degree-based* and *Centrality-based* algorithms. For example, the *Degree-based* algorithm first sorts the users based on their degree (i.e. $deg(u_i)$) and then selects the users according to that order. In the new *Degree-based* algorithm, we set the score of each user to be the degree divided by its cost (i.e. $deg(u_i)/c(u_i)$). Similarly, we build a new *Centralitybased* algorithm for comparison.

Figure 4(a), (b) and (c) show that the cost of the crowdsourced query increases when reducing the error rate threshold. However, our algorithm also

outperforms four other baseline algorithms in terms of the query cost. This nice result is attributed to the fact that we model the joint query diffusion in the proposed objective function while others treat the query diffusion of each vertex independently. The results show that our algorithm not only mines the crowdseed effectively, but is also very efficient compared with the other three baseline algorithms (Fig. 5).

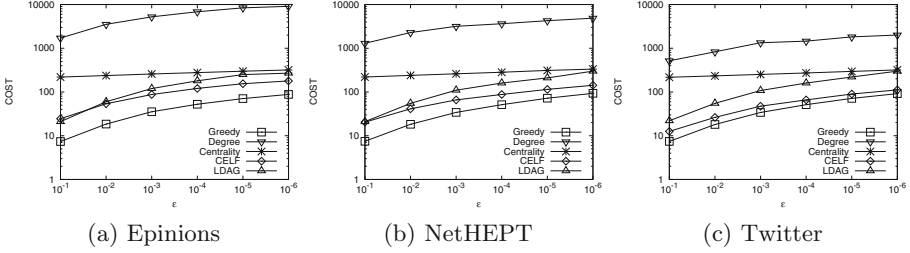


Fig. 4. Query cost v.s. error rate

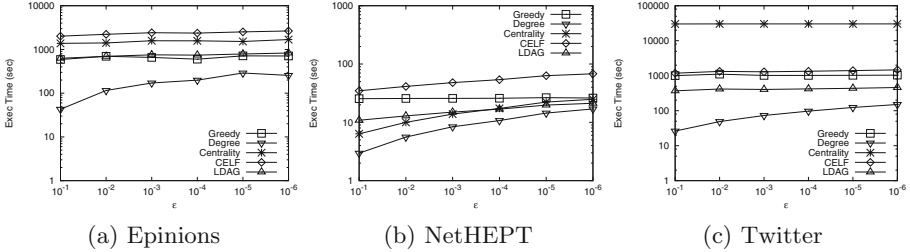


Fig. 5. Execution time v.s. error rate

7 Conclusions

We explore a new approach to processing crowdsourced queries on microblogs. Our goal is to minimize the cost of the crowdsourced query processing while the aggregated answer satisfies a specified accuracy threshold. We develop a new query diffusion model and formulate the problem of *Crowdseed Selection*. However, we prove that this problem is NP-hard and, given a crowdseed set S , the computation of query diffusion is $\#P$ -hard. We then develop a greedy algorithm to tackle the problem and a sampling algorithm to compute the query diffusion of the selected crowdseed set. We also derive an error bound for the proposed sampling algorithm. We validate the performance of our algorithm using three real

datasets. The experimental results clearly demonstrate that our algorithms are able to reduce the cost of the crowdsourced query effectively. We also show that our new incentive mechanism helps the queries to be more efficiently evaluated. As the number of smart phone users and microblog subscribers are increasing every year, our proposed approach is promising to gain further superiority over existing approaches relying mainly on a centralized platform.

References

1. Appendix. <http://www.cse.ust.hk/~wilfred/CQP.html>
2. Bozzon, A., Brambilla, M., Ceri, S.: Answering search queries with crowdsearcher. In: WWW, pp. 1009–1018 (2012)
3. Cao, C.C., She, J., Tong, Y., Chen, L.: Whom to ask?: jury selection for decision making tasks on micro-blog services. VLDB **5**(11), 1495–1506 (2012)
4. Chai, X., Vuong, B.Q., Doan, A., Naughton, J.F.: Efficiently incorporating user feedback into information extraction and integration programs. In: SIGMOD, pp. 87–100 (2009)
5. Chen, W., Yuan, Y., Zhang, L.: Scalable influence maximization in social networks under the linear threshold model. In: ICDM (2010)
6. Chen, X., Bennett, P.N., Collins-Thompson, K., Horvitz, E.: Pairwise ranking aggregation in a crowdsourced setting. In: WSDM, pp. 193–202 (2013)
7. Davidson, S.B., Khanna, S., Milo, T., Roy, S.: Using the crowd for top-k and group-by queries. In: ICDT, pp. 225–236 (2013)
8. Demartini, G., Difallah, D.E., Cudré-Mauroux, P.: Zencrowd: leveraging probabilistic reasoning and crowdsourcing techniques for large-scale entity linking. In: WWW, pp. 469–478 (2012)
9. Ghosh, S., Sharma, N., Benevenuto, F., Ganguly, N., Gummadi, K.: Cognos: crowdsourcing search for topic experts in microblogs. In: SIGIR, pp. 575–590 (2012)
10. Gomes, R.G., Welinder, P., Krause, A., Perona, P.: Crowdclustering. In: NIPS, pp. 558–566 (2011)
11. Goyal, A., Lu, W., Lakshmanan, L.V.: Celf++: optimizing the greedy algorithm for influence maximization in social networks. In: WWW, pp. 47–48 (2011)
12. Guo, S., Parameswaran, A., Garcia-Molina, H.: So who won?: dynamic max discovery with the crowd. In: SIGMO, pp. 385–396 (2012)
13. Kaplan, H., Lotosh, I., Milo, T., Novgorodov, S.: Answering planning queries with the crowd. VLDB **6**(9), 697–708 (2013)
14. Kempe, D., Kleinberg, J., Tardos, É.: Maximizing the spread of influence through a social network. In: KDD, pp. 137–146 (2003)
15. Liu, Q., Peng, J., Ihler, A.T.: Variational inference for crowdsourcing. In: NIPS, pp. 692–700 (2012)
16. Liu, X., Lu, M., Ooi, B.C., Shen, Y., Wu, S., Zhang, M.: Cdas: a crowdsourcing data analytics system. VLDB **5**(10), 1040–1051 (2012)
17. Marcus, A., Wu, E., Karger, D., Madden, S., Miller, R.: Human-powered sorts and joins. VLDB **5**(1), 13–24 (2011)
18. Parameswaran, A.G., Garcia-Molina, H., Park, H., Polyzotis, N., Ramesh, A., Widom, J.: Crowdscreen: algorithms for filtering data with humans. In: SIGMOD, pp. 361–372 (2012)
19. Parameswaran, A.G., Park, H., Garcia-Molina, H., Polyzotis, N., Widom, J.: Deco: declarative crowdsourcing. In: CIKM, pp. 1203–1212 (2012)

20. Raykar, V.C., Yu, S., Zhao, L.H., Valadez, G.H., Florin, C., Bogoni, L., Moy, L.: Learning from crowds. *JMLR* **11**, 1297–1322 (2010)
21. Selke, J., Lofi, C., Balke, W.-T.: Pushing the boundaries of crowd-enabled databases with query-driven schema expansion. *VLDB* **5**(6), 538–549 (2012)
22. Sojump. <http://www.sojump.com>
23. Trushkowsky, B., Kraska, T., Franklin, M.J., Sarkar, P.: Crowdsourced enumeration queries. In: *ICDE*, pp. 673–684 (2013)
24. Venetis, P., Garcia-Molina, H., Huang, K., Polyzotis, N.: Max algorithms in crowdsourcing environments. In: *WWW*, pp. 989–998 (2012)
25. Wang, G., Wilson, C., Zhao, X., Zhu, Y., Mohanlal, M., Zheng, H., Zhao, B.Y.: Serf and turf: crowdturfing for fun and profit. In: *WWW*, pp. 679–688 (2012)
26. Wang, J., Li, G., Kraska, T., Franklin, M.J., Feng, J.: Leveraging transitive relations for crowdsourced joins. In: *SIGMOD*, pp. 229–240 (2013)
27. Wang, X., Zhao, Z., Ng, W.: A comparative study of team formation in social networks. In: Renz, M., Shahabi, C., Zhou, X., Cheema, M.A. (eds.) *DASFAA 2015*. LNCS, vol. 9049, pp. 389–404. Springer, Heidelberg (2015)
28. Welinder, P., Branson, S., Perona, P., Belongie, S.J.: The multidimensional wisdom of crowds. In: *NIPS*, pp. 2424–2432 (2010)
29. Yi, J., Jin, R., Jain, S., Yang, T., Jain, A.K.: Semi-crowdsourced clustering: generalizing crowd labeling by robust distance metric learning. In: *NIPS*, pp. 1772–1780 (2012)
30. Zhao, Z., Cheng, J., Wei, F., Zhou, M., Ng, W., Wu, Y.: Socialtransfer: transferring social knowledge for cold-start crowdsourcing. In *CIKM*, pp. 779–788 (2014)
31. Zhao, Z., Ng, W., Zhang, Z.: Crowdseed: query processing on microblogs. In: *EDBT*, pp. 729–732 (2013)
32. Zhao, Z., Wei, F., Zhou, M., Chen, W., Ng, W.: Crowd-selection query processing in crowdsourcing databases: a task-driven approach. In: *EDBT* (2015)
33. Zhao, Z., Wei, F., Zhou, M., Ng, W.: Cold-start expert finding in community question answering via graph regularization. In: Renz, M., Shahabi, C., Zhou, X., Cheema, M.A. (eds.) *DASFAA 2015*. LNCS, vol. 9049, pp. 21–38. Springer, Heidelberg (2015)
34. Zhao, Z., Yan, D., Ng, W., Gao, S.: A transfer learning based framework of crowd-selection on twitter. In: *KDD*, pp. 1514–1517 (2013)
35. Zhao, Z., Zhang, L., He, X., Ng, W.: Expert finding for question answering via graph regularized matrix completion. *IEEE Trans. Knowl. Data Eng.* **27**, 993–1004 (2015)
36. Zhou, D., Basu, S., Mao, Y., Platt, J.C.: Learning from the wisdom of crowds by minimax entropy. In: *NIPS*, pp. 2195–2203 (2012)

Database Systems for Advanced Applications

21st International Conference, DASFAA 2016, Dallas,

TX, USA, April 16-19, 2016, Proceedings, Part I

Navathe, S.B.; Wu, W.; Shekhar, S.; Du, X.; Wang, S.X.;

Xiong, H. (Eds.)

2016, XIX, 551 p. 163 illus., Softcover

ISBN: 978-3-319-32024-3