Worker Recommendation for Q&A Services

Farbod Shahinfar
Iran University of Science and Technology

Vahid Mohseni
Iran University of Science and Technology

Ehsan Seyed Ali Akbar
Iran University of Science and Technology

ABSTRACT

Worker Recommendation is the core function of Q&A services such as Qoura and Stack Exchange. Specifically, given a set of tasks to be solved, Worker Recommendation system recommends each task to a certain group of workers whom are expected to write correct and well writen answers for given task in a short priode of time.

To address this problem we propose two algorithms. In both of these proposed algorithms a way of measuring the relevance of a worker to a problem is defined. This measurment is used for recommending a user u_i for solving a given problem p_j. In the end the top-k relevant workers are recommended by the system.

In the algorithm which is discussed first, the idea of recommending the worker who is more associated with the tags of question is formulated with a distance function. The lesser the distance is the better to recommend the user.

In the second algorithm, a graph based approch is used for modeling the problem in a fromal language and suggesting a proper set of workers for problem p_j as an answer.

At the end of this paper both proposed methods are evaluated on the data gathered from StackOverFlow question and answer website.

1. INTRODUCTION

Crowd sourcing is a mean of distributing the process of solving problems. It is also thought of as a bussiness production model. In this paradaigm, tasks are distributed among others to be done. This distribution of tasks should preformed in way that results in less cost for the bussiness. This cost, depending on the problem and company, may be time, money or other factors of interest.

A crowd sourcing process involes two groups of users, requesters and workers. A requester propose a new task to the system. One or more workers decide to work on an available task in the system. They choose a task based on their experties and interests. The solutions produced for a task are submitted to the requester for evaluation and validation. The activity history of both requester and worker is recorded for further recommendations.

This process will be efficient only if right tasks are done by appropiate workers. Moreover, there is no subset of workers to be called appropiate for all tasks in the system. That is each worker is suitable for certain types of tasks.

For huge bussiness, those with large set of workers and requesters, the rate at which a new problem is introduced to the system is fairly high. As a result of this phenomenon, each worker only notices a small subset of tasks.

To help the process of finding a proper task, this paper elaborates a system to recommend a set of workers for a proposed task. To be more specific during the paper we have considered the case of Q&A crowd source system in which tasks are problems to be answered.

The rest of the paper is structured in the following manner. Section 2 defines the problem being examined in this paper in a formal language. Section 3 suggests the first algorithm. Algorithm two is discussed in 4. Section 5 presents evaluation of our works. Section 6 and 7 are future work and conclusion respectively.

The implemented algorithms and dataset used for this paper is available online [1].

2. PROBLEM DEFINITION

In this paper we focus on presenting an efficient solution for recommending a set of workers in order to produce proper answers for a set of questions. The formal definition of this problem is as follows.

Given a set of problems $P = \{p_1, p_2, ..., p_n\}$, a set of tags $T = \{t_1, t_2, ..., t_k\}$, and a set of workers $W = \{w_1, w_2, ..., w_m\}$ our goal is to find a proper subset of workers $W' \subseteq W$ for each problem $p_j \in P$ which satisfies the conditions of having $|W'| = k$ and knowing that each worker $w \in W'$ has abilities relevant to the problem p_j.

3. ALGORITHM ONE

For solving the problem defined in section 2, in this part of the paper, an algorithm is proposed which will prodouce a proper answer.

The intuition behind this algorithm is to identify an approach for measuring closeness between a worker and a given problem. Furthermore this measure of relevance is used to discover workers who are more related to the given problem. It is supposed that a worker who is considered close the probelm is enthusiasm of the problem topic and would be able to produce a valid answer for it. We tried to keep this algorithm as naive as possible. So, the proposed algorithm is simple to trace and debug and we are sure that this algorithm is pretty correct and is trustable in results.

In the continue the algorithm is examined and its complexity, from both time and memory aspect, is analyzed.

3.1 Description

[1]Github repository: `https://github.com/vahidmohsseni/rmcourse-recommender`

In this algorithm, first, distance of each each worker ($w_i \in W$) is calculated from all problems in the given problem set ($\forall p_j \in P$). Afterward, for each problem, the top-k workeres who have the least distances to the problem are chosen.

Distance between a worker and a problem is defined as formula 1.

$$dist(w_i, p_j) = 1 - \frac{|\{t|t \in Tag(w_i) \cap Tag(p_j)\}|}{|Tag(p_j)|} \quad (1)$$

Algorithm 1 demonstrates this algorithm with its details.

Algorithm 1 This is the first algorithm

Require: W: set of workers, P: a set of problems, k: an integer and $k \leq \|W\|$
Ensure: $result$: an array of W'_j which are subset of W and $\|W'\| = k$

```
1: result[||P||] ← a new array
2: for j = 1 to ||P|| do
3:    d[||W||] ← a new linked-list
4:    for i = 1 to ||W|| do
5:       d_i ← dist(w_i, p_j)
6:    end for
7:    W' ← a new set
8:    for i = 1 to k do
9:       best_index ← argmin(d)
10:      d.remove(best_index)
11:      W'.add(W_i)
12:   end for
13:   result_i ← W'
14: end for
15: return result
```

3.2 Complexity analysis

The algorithm is comprised of a loop wich is repeated $\|P\|$ times. By the product rule it is known that the total time complexity of the algorithm will be $\|P\| \times \theta$(time complexity of loop body).

The body of the main loop is composed of two other loops and some statements. The loop from line 4 to 6 is repeated $\|W\|$ times and in each iteration the distance is calculated. next loop is repeated k times and in each repetition it search for the minimum value in an array of size $\|W\|$. As a result each iteration will cost $\|W\|$ operations, thus, time complexity of this loop is computed as $O(\|W\| \times k$.

In total the time complexity of this algroithm, is $O(\|P\| \times \|W\| \times k)$. In the given term, the $\|P\|$ is the number of problems a set of workers should be searched for. The $\|W\|$ is the total number of workers available and k is the number of workers should be recommended for each problem.

The space complexity of this algorithm is $O(\|P\| \times k + \|W\|)$.

4. ALGORITHM TWO

Although algorithm one produces correct answers, the time complexity of it, is a concern in a real-world, practical system. The number of workers of a Q&A service is much greater than the available tags and for each problem only a small subset of W has meaning full distance value. In other words our distance array for each problem is very sparse. Concluding from this observation, if an approach for pointing out the relevant group of workers for a given problem is used for filtering the workeres the time complexity would be decreased. The result of this decrease in time complexity is proportional to the complexity of the filtering approach. Algorithm two in contrast with algorithm one uses a structured body to solve the problem. This structure helps the algorithm two in efficiency which is obvious as mentioned in the next sections.

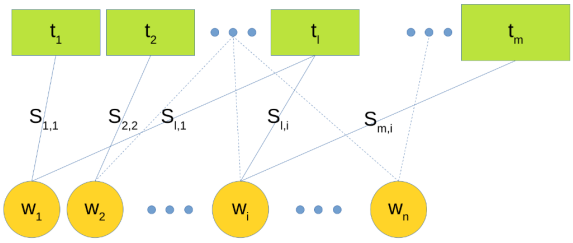

Figure 1: Structure of a bipartite speciality graph

4.1 Description

For improving the time complexity of algorithm one, this algorithm first choses a subset of W. To achieve this goal a bipartite graph is created once in pre-processing phase, before the system starts to search for the answers. Then using this data structure the search for a subset of W is perform.

The bipartite graph considered for this algorithm is called speciality graph and has two groups of nodes. A group of nodes represent available tags in the system and the other group consist of nodes representing workers. Each worker who happen to be known to have experties related to a tag, is connected to the corresponding node of that tag. Moreover this graph is weighted. Each edge connecting a tag t_l to a worker w_i has a value $S_{l,i}$ representing the score the worker has gained through solving problems of the same tag before. Figure 1 shows the general structure of this graph.

After forming this bipartite graph, the process of choosing top-k worker for each problem is as follows. The node corresponding to each tag of the given problem is probbed. For a tag t_k the workers associated with it is added to a hashmap named score_table. The score value of each chosen worker is aggregated in the score_table. After creating the score table the top-k workers having the highest aggregated score are added to W'. W' is the set containing our recommended workers.

Psuedocode in algorithm 2 depicts the approach described in this section.

4.2 Complexity analysis

Time complexity for algorithm two is $O(\|P\| \times k \times$(number of workers having one of the problem's tag)). At the worst case, in which all workers registered in the system have a connection with a tag t_l, this time complexity will be the same as the algorithm one. But as it was mentioned, because of the sparsity of relations between a worker and a tag, the score associated between t_l and all workers is zero for majority of cases. As a result, the time complexity is decreased, hence, the algorithm two will produce answers with more efficiency.

Algorithm 2 This is the second algorithm

Require: W: set of workers, P: a set of problems, k: an integer and $k \leq \|W\|$
Ensure: *result*: an array of W'_j which are subset of W and $\|W'\| = k$

```
result ← new linked-list
specialty_graph ← create_bipartite_graph()
for all p_j in P do
    score_table ← a new hashtable
    for all t_l in Tag(p_j) do
        for all w_i in specialty_graph[t_l] do
            worker_score ← specialty_graph[t_l][w_i]
            score_table[w_i] ← score_table[w_i] + worker_score
        end for
    end forW' ← a new set
    for i= 1 to k do
        worker ← argmax( score_table )
        W'.add(worker)
        score_table.remove(worker)
    end for
    result.add(W')
end for
return result
```

In contrast to time complexity, space complexity of this alogrithm is the same as the algorithm one. The specialty tree has data related to all workers so it takes space of $O(\|W\|)$. Moreover the space required by the algorithm two is $O(\|P\| \times k +$ (number of workers having one of the problem's tag)).

Viewing all aspects together, algorithm two suggests a more efficient approach for solving the problem specified in section 2.

5. EVALUATION

5.1 Data Collection

The evaluation of this work has been carried out on data gathered from StackOverFlow website. This data was collected using a crawler script which was used to extract data about problems and users. The gathered data is comprised of problems id and their correlated tags and users id along with all the tags a user has been associated with. All of the tags related to a user has a socre value. These data are stored in a sqlite file in order to be accessed easily.

5.2 Results

As it is shown in figure 2, the time needed for algorithm two to prepare an answer is reduced to less than one third of the algorithm one.

The values presented in figure 2 are average of latencies measured during testing. The problems from dataset was splitted to different sets and along with the set of all users where given to each algorithm. The k was set to 5 in all the test.

The Time complexity of both algorithm suggests that they scale linearly with respect to k parameter. The figure 3 confirms that both algorithms have linear time complexity with regard to k.

Both algorithms are tested on different size of problem set ($\|P\|$). The result of the experiments are shown in figure

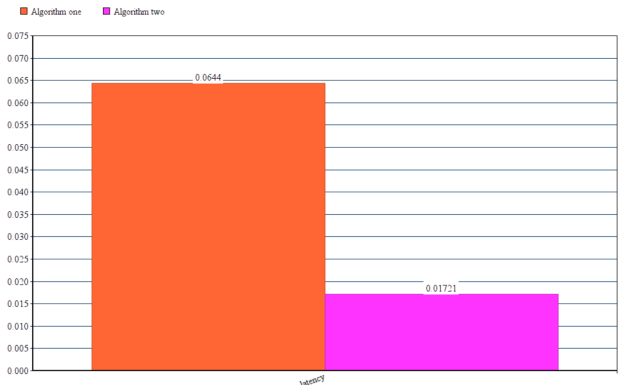

Figure 2: Comparison of latency of each algorithm

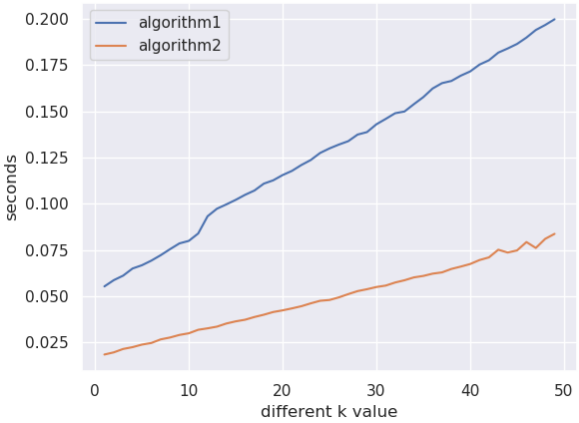

Figure 3: Clock time duration with different k values for problem sets of size 50

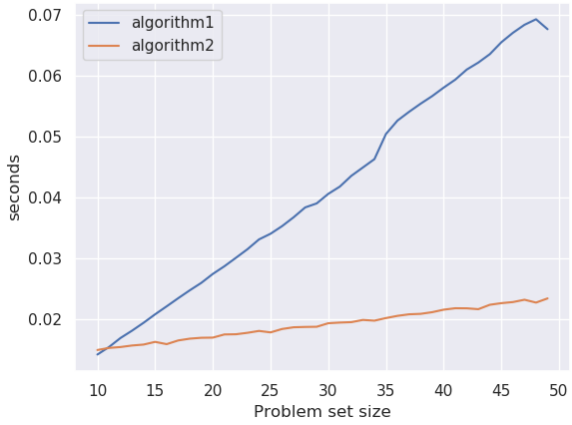

Figure 4: Clock time duration with different $\|P\|$ and $k = 5$

4. It can be concluded that both algorithms are linear with respect to the size of problem set. This conclusion complies with statements in sections 3 and 4.

6. FUTURE WORKS

In future works, we consider examining the effectiveness of our method. We believe algorithm two is more effective than the algorithm one but we do not have such a mechanism to evaluate this impression yet. Also, we are thinking about new intelligent methods such as machine learning techniques for solving this problem and compare them with these algorithms propused in this paper.

7. CONCLUSION

Crowd sourcing is a distributed approach in solving problems. It aims to reduce cost. For this system to perform efficiently, it is crucial to have a recommendation system to offer tasks to relevant workers. In this paper two algorithms are proposed for finding a subset of workers W' who are relevant to a given problem p_j. Both of these algorithms are analysed carefully. At the end these two algorithms are evaluated on a dataset gathered from StackOverFlow Q&A service.