Relative Difficulty Estimation in Community Answering Services

Student Name: Deepak Thukral
Roll Number: 2014036

BTP report submitted in partial fulfillment of the requirements
for the Degree of B.Tech. in Computer Science & Engineering

on April 18, 2018

BTP Track: Research

BTP Advisor
Dr Vikram Goyal
Dr Tanmoy Chakraborty

Indraprastha Institute of Information Technology
New Delhi

Student's Declaration

I hereby declare that the work presented in the report entitled **"Relative Difficulty Estimation in Community Answering Services"** submitted by me for the partial fulfillment of the requirements for the degree of *Bachelor of Technology* in *Computer Science & Engineering* at Indraprastha Institute of Information Technology, Delhi, is an authentic record of my work carried out under guidance of **Dr Vikram Goyal and Dr Tanmoy Chakraborty**. Due acknowledgements have been given in the report to all material used. This work has not been submitted anywhere else for the reward of any other degree.

..... **Place & Date: New Delhi, 18 April 2018**
Deepak Thukral

Certificate

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

..... **Place & Date: New Delhi, 18 April 2018**
Dr Vikram Goyal

Abstract

Automatic estimation of relative difficulty of a pair of questions is an important and challenging problem in community question answering (CQA) services. There are limited studies which addressed this problem. Past studies mostly leveraged expertise of users answering the questions and barely considered other properties of CQA services such as metadata of users and posts, temporal information and textual content. In this paper, we propose a system, a novel system that maps this problem to a network-aided edge directionality prediction problem. Given a question on a crowd sourced platform, we gauge the difficulty of the question. We used various graph models in order to model our intuition of how difficulty is associated with questions, the answerers, the asker and how over time the difficulty of one's questions change.

Keywords: Data Analysis, Stackoverflow, Graph Mining, Time-evolving networks, Network Construction, Edge Directionality Prediction

Acknowledgments

I would like to acknowledge the following

1. Dr Vikram Goyal, my advisor
2. Dr Tanmoy Chakraborty, my advisor
3. Adesh Pandey
4. Rishabh Gupta

Contents

Chapter 1

Introduction

Programmers these days often rely on various community-powered platforms such as Stack Overflow, MathWorks etc. – also known as Community Question Answering (CQA) services to resolve their queries. A user posts a question/query which later receives multiple responses. The user can then choose the best answer (and mark it as 'accepted answer') out of all the responses. In many community-based information web sites, such as Stack Overflow, users contribute content in the form of questions and answers, which allows others to learn through the contributions of the community. Such platforms have recently gained huge attention due to various features such as quick response from the contributers, quick access to the experts of different topics, succinct explanation etc. For instance, In August 2010, Stack Overflow accommodated $300k$ users and $833k$ questions; these numbers have currently jumped to $8.3m$ users and $15m$ questions posted[1]. This in turn provides tremendous opportunity to the researchers to consider such CQA services as large knowledge bases to solve various interesting problems [5, 6, 8]. We focused on questions related to java for research purposes. Our main **contributions** are:

- We propose a novel network construction technique by leveraging the user interactions and temporal information available in CQA services.
- We map the problem of 'relative difficulty estimation of questions' to an 'edge directionality prediction' problem, which, to our knowledge, is the first attempt of this kind to solve this problem.
- System turns out to be superior to the state-of-the-art methods – it not only beats the other baselines in terms of accuracy, but also appropriately responds to the training noise and handles cold start problem.
- As a by-product of the study, we generated huge CQA datasets and manually annotated a set of question pairs based on the difficult level.

For the sake of reproducible research, we have made the code publicly available at `https://github.com/deepak0004/DiffQue`.

[1] `https://stackexchange.com/sites/`

Chapter 2

Literature Survey

PageRank

PageRank is a link analysis algorithm which works on graphs to give an importance of a node in the graph. It assigns a weight to each node of a graph which is directly proportional to its importance. This algorithm is often used to rank web pages of the internet where edges represent a hyperlink between two web pages. The following is the update equation ran iteratively. d is the damping factor (usually d = 0.85). N(v) denotes neighbours of v.

$$PR(p_i) = \frac{1-d}{N} + d \sum_{p_j \in N(p_i)} \frac{PR(p_j)}{outdegree(p_j)} \tag{2.1}$$

HITS Algorithm

HITS is also a link analysis analysis on graph which is mainly used in context of web pages. It has hubs and authority scores for nodes which gets updated according to the updation rule. A good hub score represents a page that pointed to many other pages, and a good authority score represents a page that was linked by many different hubs.

The Authority score for node p is updated as follows- (Node i is connected to node p)

$$auth(p) = \sum_{i=1}^{n} hub(i) \tag{2.2}$$

The Hub score is updated as follows-

$$hub(p) = \sum_{i=1}^{n} \text{auth}(i) \tag{2.3}$$

TrueSkill

TrueSkill [2] is a Bayesian skill rating model that is developed for estimating the relative skill levels of players in games. It assumes that players skill is a normal distribution with mean m and variance v and are assigned some value at the beginning. The mean is updated as the game proceeds and it increases after a win and decreases after a loss.The amount of update will depend on its rating(mean) at the time of updating.

Chapter 3

Dataset Collection

We collected questions and answers from two different CQA services – (i) Stack Overflow[1] (SO) and Mathematics Stack Exchange[2] (MSE), both of which are extensively used by programmers or mathematicians to get their queries resolved. The former dataset was further divided into three parts – SO1, SO2 and SO3 based on the time of posting the questions. A brief description of the datasets are presented below (see Table 3.1).

Dataset	**# questions**	**# answers**	**# users**	**Time period**
SO1	100,000	289,702	60,443	Aug'08 – Dec'10
SO2	342,450	603,402	179,827	Jan'12 – Dec'13
SO3	440,464	535,416	274,421	Aug'15 – Aug'17
MSE	92,686	119,754	47,470	July'10 – Aug'17

Table 3.1: Statistics of the datasets.

Dataset	**Edge count**		
	Edge type 1	**Edge type 2**	**Edge type 3**
SO1	749,757	61,209	133,243
SO2	1,168,490	101,196	392,743
SO3	556,511	42,010	319,222
MSE	224,058	10,124	89,996

Table 3.2: Number of edges of each type in different datasets.

3.1 Test Set Generation

- We randomly selected pairs of questions from the dataset
- Three students independently and manually annotated the pair to mark the more difficult question

[1] `https://stackoverflow.com/`
[2] `https://math.stackexchange.com/`

- If all three annotated the same question as more difficult in the pair, we keep it as a ground truth AKA test set otherwise we reject the pair.

We finally created our test set consisting of 250 pairs.

Chapter 4

Related Work

4.1 Question Difficulty Estimation in Community Question Answering Services

The paper [5] aims to estimate the question difficulty in community question answering services. They used a competition-based model for estimating question difficulty by doing pairwise comparisons (competitions) between questions and users. Let question q be considered as a pseudo user u_q:

1. One competition between pseudo user u_q and asker u_a,
2. One competition between pseudo user u_q and the best answerer u_b
3. One competition between the best answerer u_b and asker u_a
4. S Competitions between the best answerer u_b and all non-best answers where S is the no of non-best answers of the question q.

4.2 Competing to Share Expertise: the Taskcn Knowledge Sharing Community

The paper [8] analyses one of the biggest Witkey websites in China, Taskcn.com. They applied social network prestige measures to a user and task networks based on competitive outcomes between them and discover the underlying properties of both users and task.

1. Suppose users A and B participate in the same task. If A wins in that task, then an edge is added from B to A.
2. If users participate in tasks X and Y. If a user A wins in task X but fails in task Y, then a edge is built from X to Y.This implies that task Y is more prestigious than task X.

4.3 Regularised Competition Model

Regularized Competition Model (**RCM**) [7] capture the significance of difficulty. It forms $\theta \in \mathcal{R}^{M+N}$, denoting the 'expert score' of pseudo users – initial M entries are expertise of users while further N are difficulty of questions. For each of the competitions, x_k vector is formed where ${x_i}^{\mathrm{k}} = 1$, ${x_j}^{\mathrm{k}} = \text{-}1$ and $y_k = 1$ if i wins over j, else $y_k = \text{-}1$. The algorithm starts at initial θ and proceeds towards negative subgradient, $\theta_{t+1} = \theta_t - \gamma_t * \nabla \mathcal{L}(\theta_t)$, where $\nabla \mathcal{L}(\theta_t)$ is the subgradient and γ_t is set as 0.001.

We consider four baselines described below; first three are existing methods and the last one is designed by us:

- **RCM:** The Regularized Competition Model proposed by Wang et al. [7].
- **Trueskill:** The approach proposed by Liu et al. [5].
- **PageRank:** The approach proposed by Yang et al. [8].
- **HITS:** We further propose a new baseline as follows – we run HITS algorithm [3] on our network and rank the questions globally based on their authoritativeness. Now given a pair of questions, we mark the one as more difficult whose authoritative score is higher.

Dataset	Underlying Network	F_1 score (%)					AUC (%)				
		RCM	Trueskill	PageRank	HITS	System	RCM	Trueskill	PageRank	HITS	System
SO1	RCM	**55.6**	54.0	62.8	52.8	33.3	**55.7**	54.1	62.8	52.9	48.1
	Trueskill	55.6	**54.0**	62.8	52.8	33.3	55.7	**54.1**	62.8	52.9	48.1
	PageRank	42.2	56.1	**50.3**	53.5	38.1	42.3	56.2	**50.5**	53.5	50.5
	System	56.5	66.6	62.1	**49.4**	**71.6**	56.6	66.9	62.4	**49.4**	**71.7**
SO2	RCM	**57.6**	52.8	51.5	49.2	32.4	**57.3**	52.9	51.6	49.3	50.0
	Trueskill	57.6	**52.8**	51.5	49.2	32.4	57.6	**52.9**	51.6	49.3	50.1
	PageRank	46.9	49.3	**49.3**	49.6	34.2	47.0	49.4	**49.4**	49.6	48.7
	System	56.9	56.1	57.2	**54.6**	**70.5**	56.9	56.2	57.3	**54.6**	**70.5**
SO3	RCM	**50.7**	52.2	54.6	48.8	36.9	**50.7**	53.2	55.4	50.1	50.0
	Trueskill	50.7	**52.2**	54.6	48.8	36.9	50.7	**53.2**	55.4	50.1	50.0
	PageRank	51.7	51.1	**50.3**	44.9	36.9	52.2	51.4	**51.0**	46.0	50.0
	System	54.1	60.7	58.9	**49.3**	**76.3**	54.7	62.2	60.7	**50.8**	**77.1**
MSE	RCM	**55.3**	58.6	55.9	52.6	32.1	**55.3**	58.6	55.9	52.6	48.7
	Trueskill	55.3	**58.6**	55.9	52.6	32.2	55.3	**58.6**	55.9	52.6	48.7
	PageRank	49.3	51.7	**53.6**	56.2	34.6	49.4	51.8	**53.6**	56.2	49.1
	System	55.6	54.8	61.6	**58.9**	**71.8**	55.7	54.8	61.6	**59.0**	**72.3**

Table 4.1: Accuracy (in terms of F_1 and AUC) of the competing methods on four different datasets. System is run with its default configuration. Boldface numbers are the accuracy of the baseline methods using the configuration reported in the original papers. Blue (red) numbers are the accuracies of the best performing (second-ranked) method. We also measure the accuracy of each competing method using the network suggested by other methods and notice that most of the methods perform better if system's network is fed into their models (which indeed shows the superiority of our network construction mechanism).

Chapter 5

DiffQue: Proposed Model

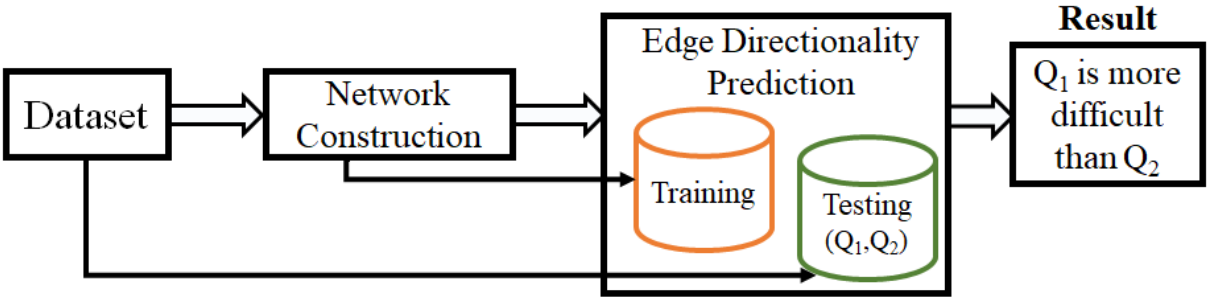

Figure 5.1: DiffQue framework

Method	Dataset			
	SO1	**SO2**	**SO3**	**MSE**
RCM	55.60	57.64	50.72	55.34
Tureskill	55.98	52.80	52.67	57.62
PageRank	47.26	49.36	50.09	50.79
HITS	45.95	54.61	49.15	54.91
DiffQue	**72.24**	**70.56**	**76.39**	**70.23**

Table 5.1: F_1 score of the competing methods on four different datasets – SO1, SO2, SO3, MSE (Baseline: RCM, Trueskill, PageRank and HITS) (Section for more details). DiffQue outperforms other baselines across all the datasets.

5.1 Network Construction

DiffQue models the entire dataset as a directed and longitudinal network $G = (V, E)$, where V indicates a set of vertices and each vertex corresponds to a question; E is a set of edges. Each edge can be of one of the following three types mentioned below.

Nomenclature: Throughout the paper, we will assume that **Bob** has correctly answered **Robin**'s question, and therefore Bob has more expertise than Robin.

Edge Type 1: An expert on a certain topic does not post trivial questions on CQA sites.

Moreover, s/he answers those questions which s/he has expertise on. We caption these two notions in Hypothesis 1.

Hypothesis 1. *If Bob correctly answers question Q asked by Robin, then the questions asked by Bob later will be considered more difficult than Q.*

Edge Type 2: It is worth noting that an edge of type 1 only assumes Bob's questions to be difficult which *will be* posted later. It does not take into account the fact that all Bob's contemporary questions (posted very recently in the past) may be difficult than Robin's current question; even if the former questions may be posted slightly before the latter question. We capture this notion in type 2 edges using Hypothesis 2.

Hypothesis 2. *If Bob correctly answers Robin's question Q, then Bob's very recent posted questions will be more difficult than Q.*

Edge Type 3: We further consider questions which are posted by a single user over time, and propose Hypothesis 3.

Hypothesis 3. *A user's expertise will increase over time, and thus the questions that s/he will ask in future will keep becoming difficult.*

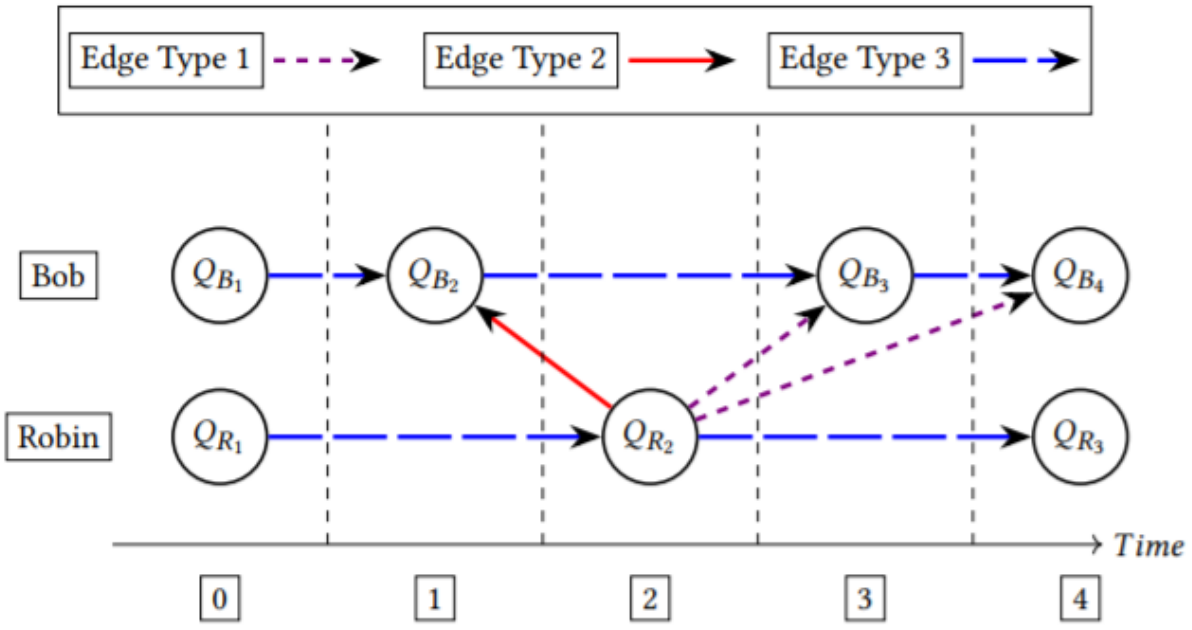

Figure 5.2: A toy example depicting the network construction of `DiffQue`. Here Bob has answered Robin's question Q_{R_2}.

5.1.1 Edge Directionality Prediction Problem

Once the network is constructed, `DiffQue` considers the 'relative question difficulty estimation' problem as an 'edge directionality prediction' problem. Since an edge connecting two questions

in a network points to the difficult question from the easy question, given a pair of questions with unknown difficulty level, the task boils down to predicting the direction of the virtual edge connecting these two questions in the network.

Our supervised model uses the following features which are broadly divided into three categories: (i) *network topology based* (F1-F6), (ii) *metadata based* (F7-F10), and (iii) *textual content based* (F11-F12)

- **[F1] Leader Follower ranking for node** a
- **[F2] Leader Follower ranking for node** b
- **[F3] PageRank of node** a
- **[F4] PageRank of node** b
- **[F5] Degree of node** a
- **[F6] Degree of node** b
- **[F7] Posting time difference between** a **and its accepted answer**
- **[F8] Posting time difference between** b **and its accepted answer**
- **[F9] Accepted answers of users who posted** a **till** T_a
- **[F10] Accepted answers of users who posted** b **till** T_b
- **[F11] Textual feature of** a
- **[F12] Textual feature of** b

In our supervised model, for each directed edge (a, b) (b is more difficult than a), we consider (a, b) as an entity in the positive class (class 1) and (b, a) as an entity in the negative class (class 2). Therefore, in the training set the size of class 1 and class 2 will be same and equal to the number of directed edges in the overall network. We use different type of classifiers, namely SVM, Decision Tree, Naive Bayes, K Nearest Neighbors and Multilayer Perceptron; among them SVM turns out to be the best model (Table 7.1(c)).

Chapter 6

Hypothesis Testing

We further conducted a thorough survey to show that three hypotheses behind our network construction mechanism are statistically significant. For this, we prepared 6 sets of edge samples, each two generated for each hypothesis testing as follows:

1. Sample 1: Choose 20 edges of type 1 randomly from the network.
2. Random Sample 1: Randomly select 20 pairs of questions (may not form any edge) which obey the time constraint mentioned in Hypothesis 1.
3. Sample 2: Choose 20 edges of type 2 randomly from the network.
4. Random Sample 2: Randomly select 20 pairs of questions such that they follow the recency constraint mentioned in Hypothesis 2.
5. Sample 3: Choose 20 edges of type 3 randomly from the network.
6. Random Sample 3: Randomly select 20 pairs of questions such that they follow the time constraint mentioned in Hypothesis 3.

Hypothesis	**Original sample**	**Random sample**	**p-value**
H1	13.25	10.44	$p < 0.01$
H2	12.12	10.11	$p < 0.01$
H3	15.40	10.66	$p < 0.01$

Table 6.1: Average number of annotators who accepted the hypotheses, and the p-value indicating the significance of our hypotheses w.r.t the null hypothesis.

Chapter 7

Results

7.1 Feature and Hypothesis Importance

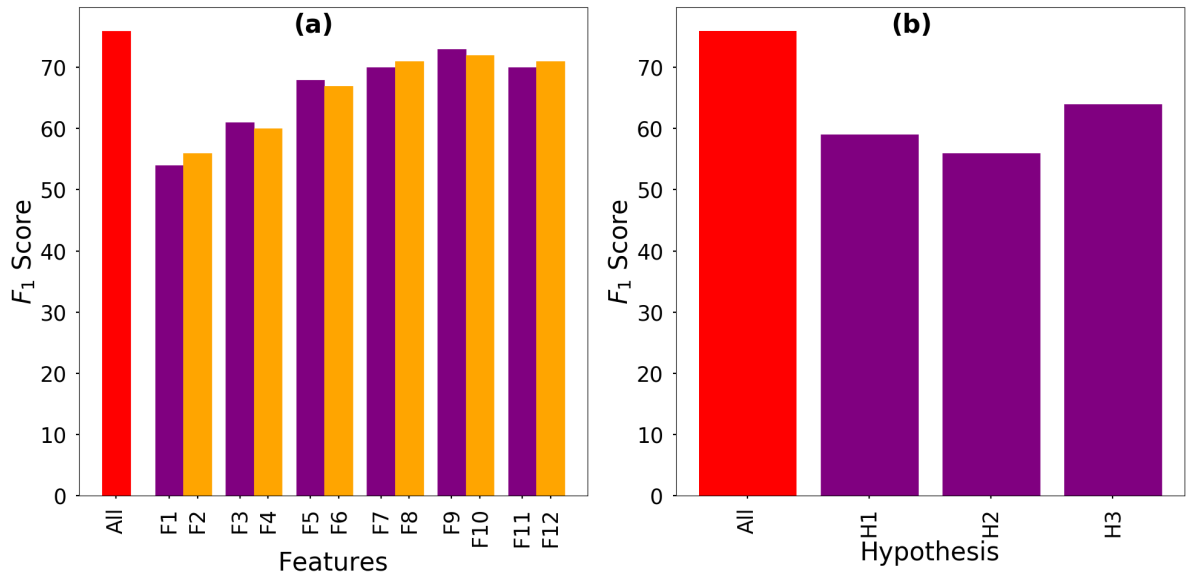

Figure 7.1: Importance of (a) features and (b) hypotheses based on F_1 score for SO3 dataset. For better comparison, we also show the results with all features and all hypotheses (All).

We also measure the importance of each feature for `DiffQue` with default configuration. For this, we drop each feature in isolation and measure the accuracy of `DiffQue`. Figure 7.1(a) shows that the maximum decrease in accuracy (27.63% decrease in F_1) is observed when we drop leader follower ranking (F1 and F2), followed by PageRank (F3 and F4) and degree (F5 and F6). However, there is no increase in accuracy if we drop any feature, indicating that all features should be considered for this task.

7.2 Parameter Selection for DiffQue

There are three important parameters of DiffQue: (i) bucket size for determining question posting time, (ii) recency of questions for type 2 edges (δ_t) and (iii) classifier for edge directionality prediction. Table 7.1 shows F_1 score of DiffQue by varying the parameter values on SO3 dataset[1].

(a) Bucket size

1	2	3	4	5	6
70.1	**76.39**	71.1	68.3	70.1	60.4

(b) δ_t

1	2	3	4	5	6
76.39	76.11	75.88	75.88	75.91	76.2

(c) Classifier

SVM	DT	NB	KNN	MLP
76.39	63.9	75.2	69.1	75.4

Table 7.1: F_1 score of DiffQue with different parameter combination on SO3 dataset. For each parameter, we vary its value keeping the other parameters default.

7.3 Capability of Domain Adaptation

		Testing			
		SO1	**SO2**	**SO3**	**MSE**
Training	**SO1**	71.65 (55.60)	66.99 (52.65)	76.68 (48.88)	71.49 (56.92)
	SO2	70.21 (60.09)	70.53 (57.60)	77.24 (54.60)	67.60 (54.15)
	SO3	70.27 (56.15)	66.65 (50.31)	76.39 (50.70)	71.22 (53.35)
	MSE	70.98 (51.60)	66.99 (49.99)	76.36 (53.28)	71.84 (55.30)

Table 7.2: F_1 score of DiffQue and RCM (within parenthesis) for different combination of training and test sets. Results of other baselines are provided in Supplementary [1].

7.4 Handling Cold Start Problem

DiffQue handles the cold start problem by exploiting the textual description of questions. To handle the problem, we run Doc2Vec [4], a standard embedding technique on textual description of all the questions and return, for each of Q_1 and Q_2, k most similar questions (based on cosine similarity) which are not brand-new and then find difficulty by majority rule (how many previous question of Q_1 beat Q_2.

To test the efficiency of our cold start module, we remove 50 annotated pairs (edges and their associated nodes) randomly from the network. These pairs form the test set for the cold start problem. Figure 7.2 shows that with the increase of k, DiffQue always performs better than RCM (the only baseline which claimed to handle cold start problem), indicating DiffQue's superiority in tackling cold start problem.

[1]The pattern was same for the other datasets and therefore not reported here.

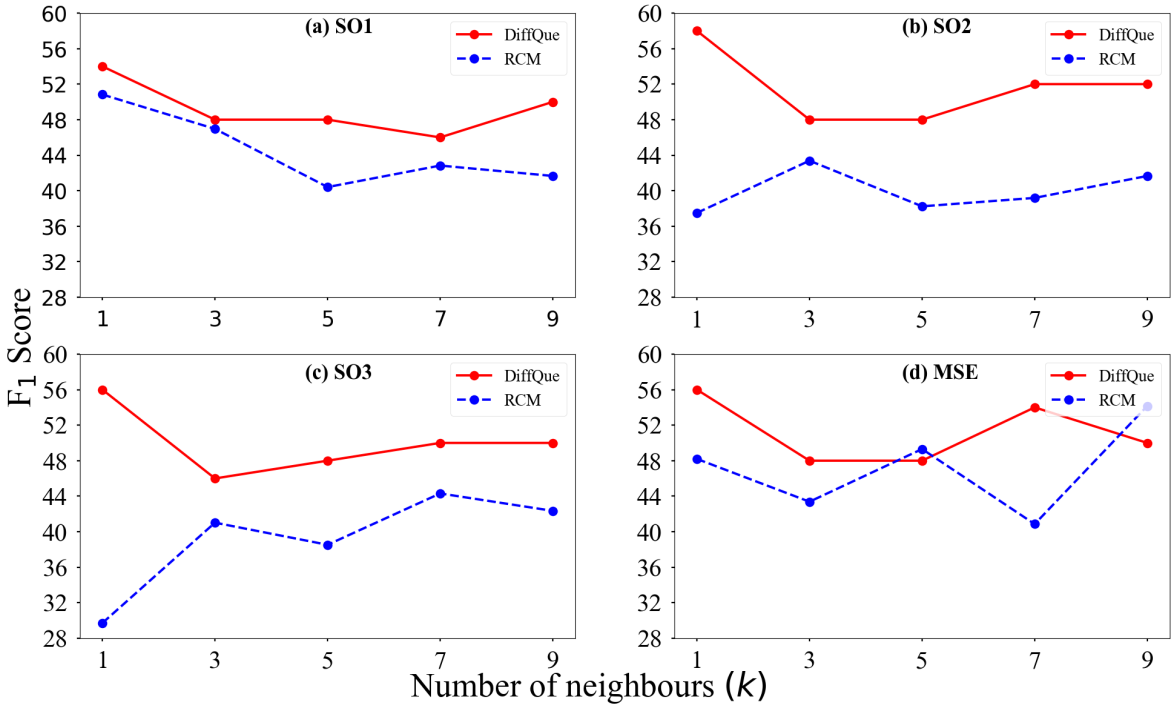

Figure 7.2: Accuracy of `DiffQue` and RCM with different number of neighbors.

Chapter 8

Conclusion

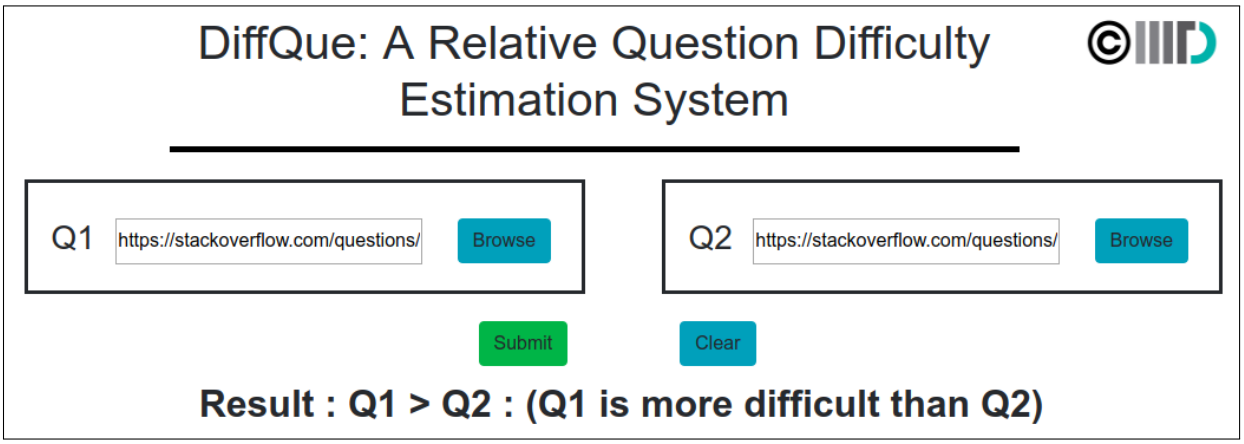

Figure 8.1: User interface of `DiffQue`.

We have designed an experimental version of `DiffQue` . Figure 8.1 shows the user interface of `DiffQue`.

In this paper, we proposed `DiffQue` to address the problem of estimating relative difficulty of a pair of questions in CQA services. `DiffQue` leverages a novel network structure and estimates the relative difficulty of questions by running a supervised edge directionality prediction model. `DiffQue` turned out to be highly efficient than four state-of-the-art baselines w.r.t. the accuracy, robustness and capability of handling cold state problem.

Bibliography

[1] Supplementary. `https://www.iiitd.edu.in/~vikram/DiffqueSI.pdf`, 2018.

[2] Herbrich, R., Minka, T., and Graepel, T. Trueskill(tm): a bayesian skill rating system. In *NIPS* (2007), pp. 569–576.

[3] Kleinberg, J. M. Authoritative sources in a hyperlinked environment. *JACM 46*, 5 (1999), 604–632.

[4] Le, Q., and Mikolov, T. Distributed representations of sentences and documents. In *ICML* (2014), pp. 1188–1196.

[5] Liu, J., Wang, Q., Lin, C.-Y., and Hon, H.-W. Question difficulty estimation in community question answering services. In *EMNLP* (2013), ACL, pp. 85–90.

[6] Vasilescu, B., Serebrenik, A., Devanbu, P., and Filkov, V. How social q&a sites are changing knowledge sharing in open source software communities. In *ACM CSCW* (New York, NY, USA, 2014), pp. 342–354.

[7] Wang, Q., Liu, J., Wang, B., and Guo, L. A regularized competition model for question difficulty estimation in community question answering services. In *EMNLP* (2014), ACL, pp. 1115–1126.

[8] Yang, J., Adamic, L. A., and Ackerman, M. S. Competing to share expertise: The taskcn knowledge sharing community. In *ICWSM* (2008), pp. 161–168.