Cross-Platform Expertise Characterization for Question Routing

Master's Thesis

Aditi Rawat

Cross-Platform Expertise Characterization for Question Routing

THESIS

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE
TRACK SOFTWARE TECHNOLOGY

by

Aditi Rawat
born in Agra, India

Web Information Systems
Department of Software Technology
Faculty EEMCS, Delft University of Technology
Delft, the Netherlands
http://wis.ewi.tudelft.nl

Cross-Platform Expertise Characterization for Question Routing

Author:	Aditi Rawat
Student id:	4518187
Email:	A.rawat@student.tudelft.nl

Abstract

Community Question-Answer(CQA) services are online portals with a community of people that share similar interests, and post question-answers for solving the difficulties they have. These services have huge social impact and have become a place where knowledge is created and distributed. Despite the advances in Question Routing and Recommendation Systems, CQA still struggles in handling large number of unanswered questions and suffer with the quality of answers provided in general. An effective solution to tackle this problem is to consider the expertise of users when routing a question to its potential answerers. There are many existing methods for measuring expertise but they tend to favour more active users rather than expert users. To address this problem, we look into cross-platform behaviour of a user and incorporate his cross-platform expertise. We provide insights into performance of question routing with this method.

We perform an extensive literature survey about the CQA system, question routing, recommendation systems and types of expertise, to map a users' cross-platform behavior with a certain type of expertise. We propose a methodology to match users across multiple platforms, with which we measure expertise according to mapped user behaviour on those platforms. Finally we estimate the reliability of expertise measurement when applied to question routing. We study and discuss the effect of parameters such as division of data into training and test set sizes and use factorization machine for recommendation system.

Preface

|| *Om Shri Ganeshay Namah* ||

This report is the final result of my Master's thesis project, marking the end of my time as a computer science student at Delft University of Technology after two amazing years. While the project had its ups and downs and was challenging from start to finish, I am proud of what I was able to achieve. Before I leave the university to step into the corporate world, I would like to take this opportunity to thank several people.

To start, I would like to thank Dr. Alessandro Bozzon, my supervisor during the project, whose guidance and feedback always pointed me in the right direction and motivated me to always think professionally and one step further. I would like to thank Dr. Geert-Jan Houben and Dr. Venkatesha Prasad, the other two members of my thesis committee, for their efforts and feedback. I also thank my uncle and aunt for providing constant support and for proofreading early drafts of my thesis. A very special thank you to Jie Yang, whose support, insightfulness and unfailing willingness to provide help when I needed it was incredibly useful.

Apart from the people I worked with during my thesis project, I would like to thank all of my friends, especially Rucha, Sambit, and Rajat, for motivating me in my lows and providing very important insights at all crucial moments. I feel truly privileged for having you all in my life.

Finally, I would like to thank my parents and God for supporting me in everything I do and always believing in me.

Aditi Rawat
Delft, the Netherlands
October 12, 2017

Contents

List of Figures

List of Tables

Chapter 1

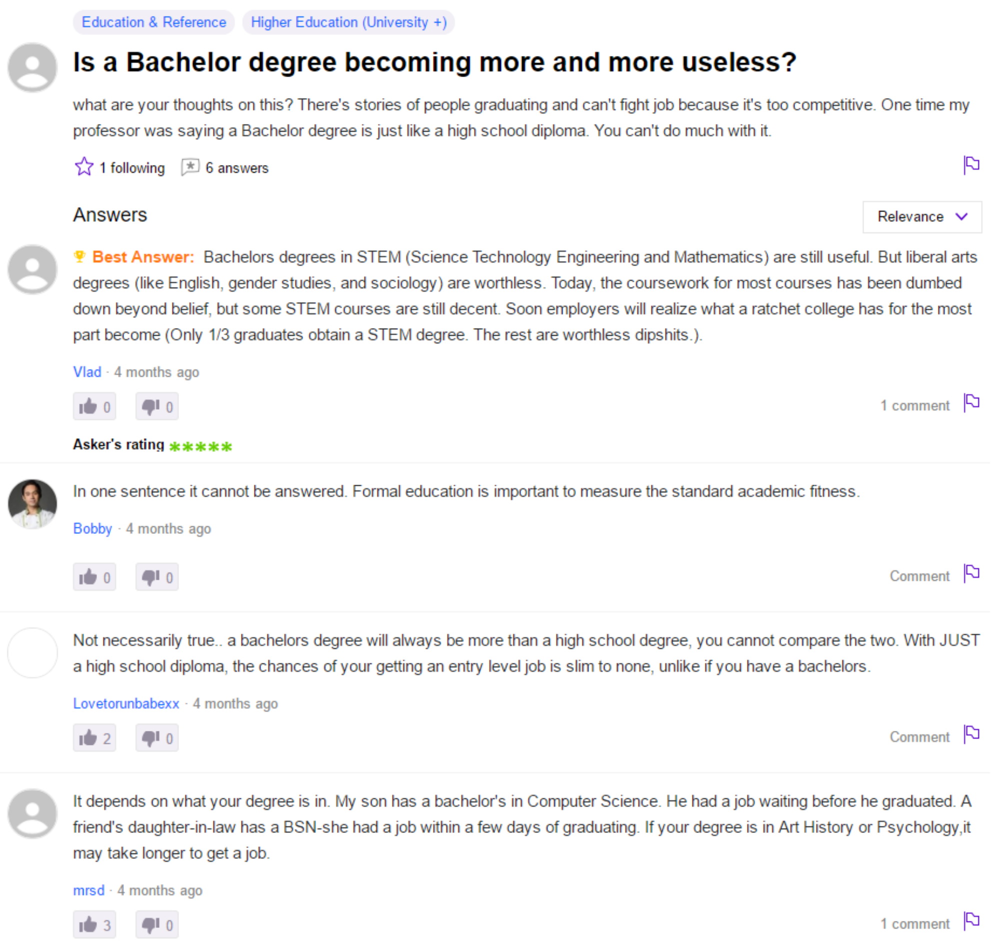
Introduction

1.1 Introduction

Community question answering (CQA) services are online portals with a community of people that share similar interests. It is a service that contributes to solving many difficult questions we have, by retrieving millions of questions and answers posted by people active on these services. The importance and huge societal impact of such services are evidenced by the heavy traffic observed[32] on popular CQA sites like Yahoo Answers [1], Baidu Zhidao [2], and Stack-Overflow [3]. Previously questions would be sent or directed to one person, or multiple people at a time. In a CQA system, questions are placed on the online platform and can be answered by anyone who has access to the platform and thus are not directed to a single person or multiple people. An example of a question posted on Yahoo Answers is given in Figure 1.1, where there are five answers for the question. The advantage of Community Question and Answering platforms is that knowledge is created and distributed. Users that were previously unable to find a person who could answer their question now have a sea of information available to find the answer. In the case where an answer is not found by traditional means, a CQA platform offers the asker a much wider audience to ask their question.

In general, on a CQA platform, a person (the asker) posts a question and waits for answers from other users (the answerers). The users search for questions that are of their potential interest, and answer them. Other users give up-vote or down-vote to a question/answer if he/she finds the question/answer appropriate or irrelevant/not up to the mark, respectively. To encourage participation, question answer platforms employ effective gamification mechanisms[4] that motivate users by showing a public *reputation score* (calculated by summing the number of preferences obtained by all the posted questions and answers), and by assigning badges after achieving pre-defined goals (e.g. achieve a score of 100 or more for an answer, question score of 100 or more). These badges and votes act as an incentive for users to be more active on the platform.

[1] answers.yahoo.com
[2] zhidao.baidu.com
[3] stackoverflow.com


Education & Reference | Higher Education (University +)

Is a Bachelor degree becoming more and more useless?

what are your thoughts on this? There's stories of people graduating and can't fight job because it's too competitive. One time my professor was saying a Bachelor degree is just like a high school diploma. You can't do much with it.

1 following 6 answers

Answers Relevance

Best Answer: Bachelors degrees in STEM (Science Technology Engineering and Mathematics) are still useful. But liberal arts degrees (like English, gender studies, and sociology) are worthless. Today, the coursework for most courses has been dumbed down beyond belief, but some STEM courses are still decent. Soon employers will realize what a ratchet college has for the most part become (Only 1/3 graduates obtain a STEM degree. The rest are worthless dipshits.).

Vlad · 4 months ago

0 0 1 comment

Asker's rating ★★★★★

In one sentence it cannot be answered. Formal education is important to measure the standard academic fitness.

Bobby · 4 months ago

0 0 Comment

Not necessarily true.. a bachelors degree will always be more than a high school degree, you cannot compare the two. With JUST a high school diploma, the chances of your getting an entry level job is slim to none, unlike if you have a bachelors.

Lovetorunbabexx · 4 months ago

2 0 Comment

It depends on what your degree is in. My son has a bachelor's in Computer Science. He had a job waiting before he graduated. A friend's daughter-in-law has a BSN-she had a job within a few days of graduating. If your degree is in Art History or Psychology,it may take longer to get a job.

mrsd · 4 months ago

3 0 1 comment

Figure 1.1: A question with 5 answers on the CQA Yahoo Answers. *One answer is hidden.*

1.2 Problem Statement

In current CQA services, a user who submits his question is required to either wait for other users to post answers to the question, this may take several days and sometimes results in incorrect or unwanted answers, or the other way is the use of previous history of CQA sites. Furthermore the previous history often contains some restricted answer sets and the user has to deal with the difference in formulating the questions. The user might formulate the question in a certain way and a similar question might be formulated in a different way in the previous history. Another problem with the community question answer system is the low participation rate of the users. That is only small portion of users are responsible for answering a large number of questions. Two main reasons for low participation are that, the users are not willing to answer the question or are not experts and the second that the users are not aware of new questions been posted that are of their interest.

The undirected question asking mechanism has put a burden on the answerer who now must choose from the multitude of questions available in the platform[29]. This burden on the answerer has two consequences. First of all, there will be questions that remain unanswered, as there are typically a lot of questions but so few answerers available. Second the quality of the answers may suffer, as some answerers want to maximize the number of answered questions, and not the correctness and soundness of the explanation. Several studies[50] show Q/A platforms are fuelled by a set of highly active users that alone contribute to the vast majority of the produced content. They do so just to increase their reputation (which is directly proportional to the activeness of the user), and not pay attention on the quality of answers they provide. Such users are called Sparrows[50]. The other category of users called Owls[50] are the ones who write good quality answers, but they are not very active and answer very few questions as compared to the Sparrows[50]. Furthermore, users also have to wait for long period of time to receive an answer. Both cases degrade the effectiveness of the knowledge generation in the CQAs, which is their primary goal.

To help creating a better knowledge base, the burden on the answerer should be reduced. One way of doing this is to suggest questions to candidate answerers such that the likelihood of obtaining a good answer is maximized. This process is called Question Routing (QR). When a question enters a Question Routing system, the system predicts which users are most likely to give a good answer to the given question. To achieve this goal the system creates a profile for all users, which leverages information such as the number of answers given by the user and the users' preference for certain topics to estimate how likely a user will answer a given question. Almost all approaches in Question Routing consider the question content and user preferences for the question and user profile. Using only those features ignores one crucial aspect that exists in non-CQA question and answering; the knowledge level of the answerer, which might be reflected by the quality of a user's answers. Currently few approaches consider the quality of answers provided by a user. Both a novice and an expert may have the same preference, but the level of knowledge is quite different. The evaluation of the knowledge level, or expertise, is currently not considered in many Question Routing approaches. Furthermore, the expertise being studied in these works, measures the levels of expertise based on the previous question answering history. However in general, a persons knowledge can not be assessed by just one source of information. There is a need to go across platform to understand a users' actual expertise in a topic. Considering expertise inferred from multiple sources is a natural extension of the currently existing approaches. Expertise will add extra information in the Question Routing system which would allow it to make a more informed decision. The working hypotheses is that including information about candidate answerer expertise measured from across platforms can improve the performance of QR systems.

1.3 Research objectives

The motivation and idea behind my thesis is to improve the expertise characterization within the community question and answer system. We aim to build a recommendation model that recommends users based on their expertise for a particular question within a topic. In this thesis, we propose that, when a question is supplied, the question answer system selects potential responders from which to solicit responses. Having said that, the core focus of our thesis is to study the metrics of expertise measurement. The objective of this thesis can be summarized in the following research question,

RQ: How can *expertise* be characterized from data across web collaborative platforms to improve the performance of Question Answering System?

This question can further be sub-divided into multiple questions that need to be addressed in order to solve our problem.

- **RQ1:** What are the different types of expertise and how can they be measured?

  Objective 1: To conduct a literature review about expertise characterization.

  Here we aim to carry out a literature survey about expertise in general, expertise characterization currently incorporated in the CQA systems and expertise characterization in a cross-platform environment.

- **RQ2:** How can a user be modelled using data from multiple web collaborative platforms?

  Objective 2: To design user profile from data across web collaborative platforms.

  We first aim to choose web platforms that can be used for user modeling. Then we gather data from them and understand how can the user profiles be linked to create one user profile consisting of data from all platforms.

- **RQ3:** Can user modelling based on expertise evaluation improve the performance of question and answer routing systems?

  Objective 3: To identify metrics for expertise evaluation and use it to evaluate question answer routing system.

  Using the data created from user profiles, identify metrics for expertise evaluation. Then use these metrics to evaluate the performance of the recommendation system of question answer routing service.

Previous works [43, 49, 22] suggested ways to approach our research questions. We used and extended the methods described in [43] to create our user base. We extended the user actions described in [49] to map the user behavior with expertise type. Our work is a continuation of previous works and takes us closer in solving problems addressed in previous section.

In order to find answers to second and third research question, first we needed to build a user base upon which further analysis could be done. For that, from [49], programmers ask and answer software engineering related questions on Stack-Overflow, collaboratively code on Github, and share software related content on Twitter. The user activity on Stack-Overflow alone gives an incomplete picture of user expertise. In order to study and find various types of expertise, we created a user base of users from Stack Overflow, Github and Twitter. Users on Stack-Overflow post question within the domain of software development. They tag questions which depicts a particular topic that questions falls into. Users who answer these questions provide links to their Github repositories, or give codes within the answers. On Twitter, these users interact socially and share their knowledge with other users. Thus for our study, we chose to use the user base of Github, Twitter and Stack-overflow as a basis for our research, as suggested in [49].

Next we needed to create an expert recommendation model based on this user base. We first study various recommender systems and then see which will be applicable on our study. We then perform few experiments to find the better performing recommender model. Our experiments include fine tuning the hyper-parameters and then analyzing the performance of recommender models using different auxiliary features. We compare and evaluate the results to find the better performing recommendation model to create the recommendation system for question routing.

1.4 Contributions

By automating the process of finding an expert for a question, the expert user will be able to find relevant questions to answer, and the questions will get high quality answers, thereby reducing the cost of investment of time required by an expert to find suitable question, and improve the overall productivity of question answer system.

The contribution provided by us in this piece of work include :

- Literature survey of expertise characterization for question answer routing.
- Data Collection Pipeline from multiple web platforms to build user profiles.
- Framework of expertise characterization using data from multiple web collaborative platforms.
- Calculating cross platform expertise using data from web collaborative platforms.
- Evaluation of performance of recommender systems for question routing with different configurations.

1.5 Thesis Outline

This thesis is divided into 7 chapters. First, in Chapter 2, we describe the background and related work on expertise characterization for community question and answer systems. Then, in Chapter 3, we present conceptual framework of the whole system and data collection pipeline to create user profiles. In chapter 4, we present the cross-platform metrics to calculate expertise in CQA systems. In Chapter 5, we describe the experimental setup to evaluate the performance of systems. In Chapter 6, we present and discuss the results. Lastly in Chapter 7, we conclude the thesis by discussing the future scope of this thesis work.

Chapter 2

Background and Related work

In this chapter, we will review some of the research work done on community question answering, question routing and factorization machines. We also look into previous work done on expertise recommendation. This acts as a base for our answer to RQ1. RQ1 is answered in chapter 4

2.1 Community Question Answering

In the past few years, Community Question Answering websites such as Yahoo answers, Quora and Stack-overflow have been building very large archives of questions and their answers [2, 19, 40, 35]. Research on community question answering has seen a significant growth. Research on CQA was mostly studied on two problems.

- Decrease the waiting time for a personal response
- Cater to the problem of large number of unanswered questions

One of the main goals of the below mentioned researches is to decrease the waiting time for a personal response. Relying on the available archive, one can approach this problem by either finding similar questions or relevant answers. Relying only on the questions available in the archive, the objective is to find similar previously answered questions by the QA community.

The problem of binding similar previously answered questions or question recommendation is tackled in [12] by representing questions as graphs of topic terms, and then ranking recommendations on the basis of these graphs. An automatic method for finding questions that have the same meaning is proposed in [27]. This method finds semantically similar questions that have little word overlap. Including the answers available in the archive, the main purpose is to find a right answer in the QA archive for a given question.

To build an answer finding system, four statistical techniques are used in [11] including TF-IDF, adaptive TF-IDF, query expansion and statistical translation. A semantic knowledge base (WordNet) is used in [26] to improve the ability of classical information retrieval approaches in matching questions and answers. Additionally, non textual features are used to improve the answer search quality in [28].

The second major problem of CQA caters to the problem of large number of unanswered questions. The study in [5] present an algorithm ENTITY-ALCHEMY to reduce the number of unanswered questions in question categories with high entity usage. According to the authors, reusing past resolved questions is an effective method for reducing the number of unanswered questions in a CQA system. This paper showed that in question categories with a lot of named entities and entity name variations, using knowledge base information and applying entity linking to identify and disambiguate named entities finds most of the similar past resolved questions to a given question.

Peer production is a term used to describe the phenomenon of distributed users collaborating to contribute value to others without oversight or management by a business enterprise. In peer production systems users, are motivated by three types of rewards: monetary, intrinsic hedonic, and social-psychological [10]. Many instances of peer production forgo monetary rewards completely, such as Wikipedia and the series of Games with a Purpose [1]. These games intrinsically reward volunteer users for performing tasks such as image tagging. Not all peer production systems are entirely benevolent; Pouwelse et al. explain that many systems contain Pirates and Samaritans [39]. Pirates may add value by illegally sharing content at the expense of the content creators.

Question and answer forums are another form of peer production, where volunteer users add value by answering questions posed by others. It is possible for a pirate to intentionally supply low quality answers, but most mechanisms driving QA forums do not reward this behavior. Whereas Google Answers provided a monetary incentive to answer questions, Yahoo! Answers(YA) relies on only intrinsic and social rewards. Adamic et al. have taken a close look at user behavior and content in YA [2]. They have characterized the distribution of question topics and demonstrated that responders who primarily respond to a small set of related topics have expertise in that topic and are therefore more likely to provide answers that are rated highly. The incentive mechanism used by YA is point-based; points are a non monetary reward and they are assigned according to user behaviors such as asking a question, answering a question and providing an answer that is selected as the best answer.

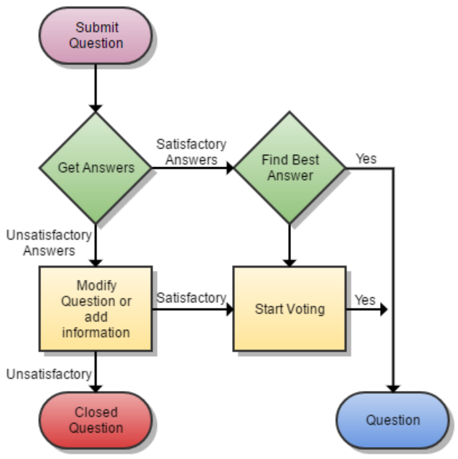


Figure 2.1: The questioning process of Stack-Overflow

Figure 2.1 shows the question answering process of stack-overflow as an example of QA routing. The registered users on the portal submit a question. Other users who find this question, provide answers. If the owner of the question finds the answer satisfactory, he votes the answer and chooses the best answer. If the answers are unsatisfactory, or requires additional information, the owner, modifies his question. If their is no relevant answer, the community decides to close the question. Questions that have also found their best answers are considered as closed questions.

Jain et al. have performed a detailed analysis on the YA incentive mechanism and proven that the best answer scoring rule does not always reward responders appropriately and they suggest approval voting or asker-distributed-the-points rules [25]. The decision-theoretic framework for QA forums presented here takes a different approach for motivating responses. Like other QA systems it assumes that responders gain some intrinsic or social reward for participating, but the framework is designed to lower the responders time investment and increase the questioners confidence in the provided answers. The core of the design framework is a recommender system. This recommender selects potential responders to answer a given question. This lowers responders time investment because they do not need to search through a list of questions to find one that they are willing and capable of answering. Moreover, this recommendation system increases questioner confidence in the responses because the recommendation algorithm is designed to identify the most appropriate experts (responders) for a question.

The above study proves a major improvement in routing questions to expert users. However, they calculate the expertise only based on the previous answering history. As discussed above, in order to reduce the number of unanswered questions, we need to determine the expertise of a user, and route a relevant question to him. Further in the chapter we will discuss some more studies done to measure expertise of a user.

2.2 Recommender System

Recommender Systems form the basis for Question Routing. The techniques used in both fields are very similar, but their purpose is different. In Recommender Systems the focus is on recommending items to users to aid the user in his quest for new and interesting content. In Question Routing the focus is on finding users that are able to answer a given question.

Even though Recommender Systems and Question Routing techniques can be similar, not all types of Recommender Systems can be used for Question Routing. This section will discuss which types are and which aren't fit for use in Question Routing.

Recommender Systems are systems that recommend an item to a user that interacts with the system. Recommendation Systems can be classified into two classes. One class of recommendation system is content-based filtering (CBF); it measures similarity by looking for common features of the items. A second class of recommendation system uses collaborative filtering (CF); these measure similarity of users by their item preferences and/or measure similarity of items by the users who like them.

Type	Description	Implicit	Explicit
Unary	Only preference or non-preference is expressed	click, buy, answer	favourite, like
Binary	Both preference and non-preference are expressed		like/ignore, thumbs up/down
Discrete	Preference is expressed as a value in a discrete range		Star rating, grade
Continues	Preferences are expressed on a continue scale		Slider green/red, slider dislike/like

Table 2.1: Different types of preference expression in Recommender Systems

2.2.1 Preference Expression

Both Content-Based Filtering and Collaborative Filtering depend on how users express their preference for an item. Research has shown that different types of expressed preference exists in Recommender Systems [36]. For example clicking on something is an implicitly expressed preference, while liking something is an explicitly expressed preference. In explicit expression the users consciously expresses their preferences. In implicit expression user preferences are derived from user actions. The two main categories as described in [22] are listed in Table 2.1

2.2.2 Content-based Filtering

In Content Based Filtering (CBF) items are modeled by their properties $p_1....p_n$ [36] and users are modeled by their preferences. Item recommendation is done by finding the items that have properties which matches with the users' preferences. The best matches are returned as recommendations for the user. An overview on how this process works is given in Figure 2.2

In a CBF Recommender System, finding matches for a user u can be done in a multitude of ways. Whichever way is chosen, comparing a user with an item is a required step. The items that are best comparable to the user u are provided as recommendations. In most cases the user u and item i are compared based on their similarity [36]. In the case both users and items are represented by the same Vector Space Model (VSM) m, the similarity between u and i is often based on the cosine similarity which represents the closeness of two vectors in space. If two vectors are aligned in their vector space, the value of the cosine similarity is higher then when they are not aligned.

$$cosinesimilarity(u,i) = \frac{\sum_k^{|m|} m_{ki}.m_{ku}}{\sqrt{\sum_k^{|m|} m_{ki}^2}.\sqrt{\sum_k^{|m|} m_{ku}^2}} \tag{2.1}$$

2.2.3 Collaborative Filtering

Collaborative Filtering (CF) recommends items based on expressed user preferences [41]. The user preference is expressed either implicitly or explicitly. The expressed preference is used to guess the preference of a user for the unseen item. Figure 2.3

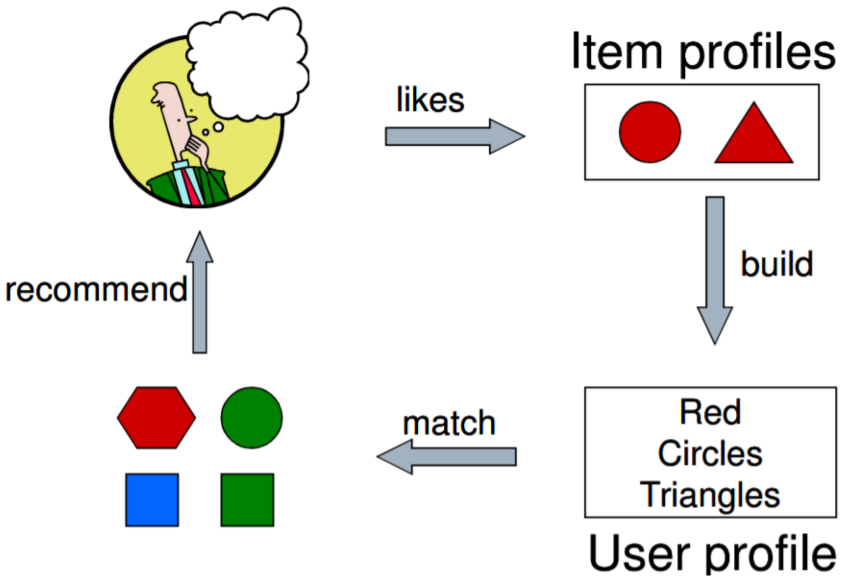


Figure 2.2: Workflow of CBF Recommendation[46]. *In this workflow, the user profile is created based on expressed preferences on certain features. The item profile is created based on analyzing its features. Matching user feature preferences and items features results in a set of items that are recommended.*

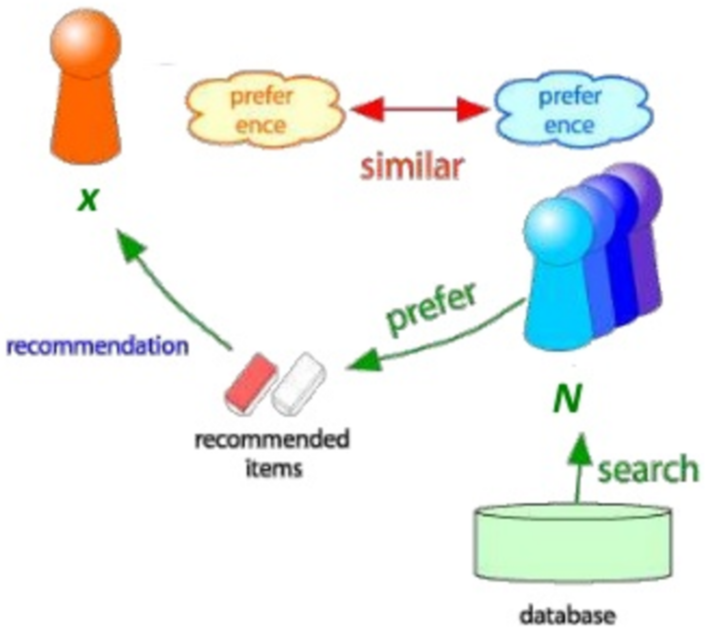


Figure 2.3: Work-flow of CF Recommendation[46]. *In this work-flow, the X users' profile is created based on expressed preferences. Then we find a set of N users whose preferences are similar to user X's preferences. Then estimate user X's ratings based on ratings of users in N. Highest rated items are recommended to user X.*

shows the work-flow of collaborative filtering. Two strategies exist that utilize expressed user preferences to generate recommendations. The first is a user-based approach, where users are compared to each-other to generate a recommendation set for a given user. The other approach is the item-based approach where items are compared to each-other to generate a recommendation set for a given user. Both approaches are explained in this section.

User-based

In user-based Collaborative Filtering, similarity is calculated between users based on the similarity of their preferences. If user u and user u_2 both have rated items $i_{a,b,c}$ the same, but user u_2 has rated i_d as well, i_d may be interesting for u_1 as well. In a user-based CF recommender system the set I of all items that are unknown for a given user u are considered. For each item i in I every user u_i in the set of users U_i that have expresses their preference for i is compared for similarity with user u. The predicted preference of user u for i is the average of the ratings given by the other users, weighted by the similarity of u and u_i. The items with the highest predicted preference for user u are given as suggestion. An example of user based CF is shown in figure 2.4.

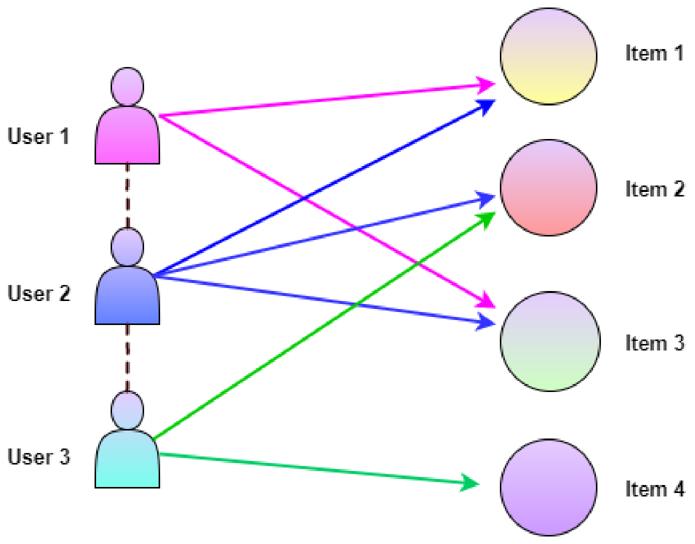


Figure 2.4: Example of user based Collaborative Filtering. *In this example unary preferences are given, which are represented by solid lines between the figures(users) and the circles(items). User-user interaction is shown by dotted lines. Recommendation for user 1 can only be done using item 1.*

In user-based CF recommender systems existing preferences for an item of other users are required in order to predict the preferences of a user for this item. This requirement is inherent for this type of recommendation, no similarity can be calculated for a user that has no expressed preference and no preference can be predicted for an item that has no expressed preferences yet. The inability to recommend new users/items is known as the 'Cold Start Problem' [42]. Using this approach for Question Routing poses a serious problem, because QR specifically routes new and unrated questions.

Item-based

In item-based Collaborative Filtering similarity is calculated between items. The item similarity uses expressed user preference to calculate a similarity value [41]. Users that have interacted with both item i and j are used in some similarity calculation $S(i,j)$ between item i and j. Which similarity calculation is used differs per type of expressed preferences of a user. For example when preferences are unary, the similarity between two items can be calculated by the Tanimoto[24] similarity as given in equation 2.2

$$S(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|} \tag{2.2}$$

U_i and U_j are the sets of users that have respectively expressed their preference for item i and item j. Thus the more users have expressed their preference on both items, the higher the similarity is between the items. Item-based CF suffers from the cold start problem as well, new items cannot be recommended because no users have expressed a preference for this item. An example of the item-based CF works is depicted in Figure 2.5. The cold start problems also exists for new users. With no preference for an item, no items can be compared, thus no items can be suggested to a user.

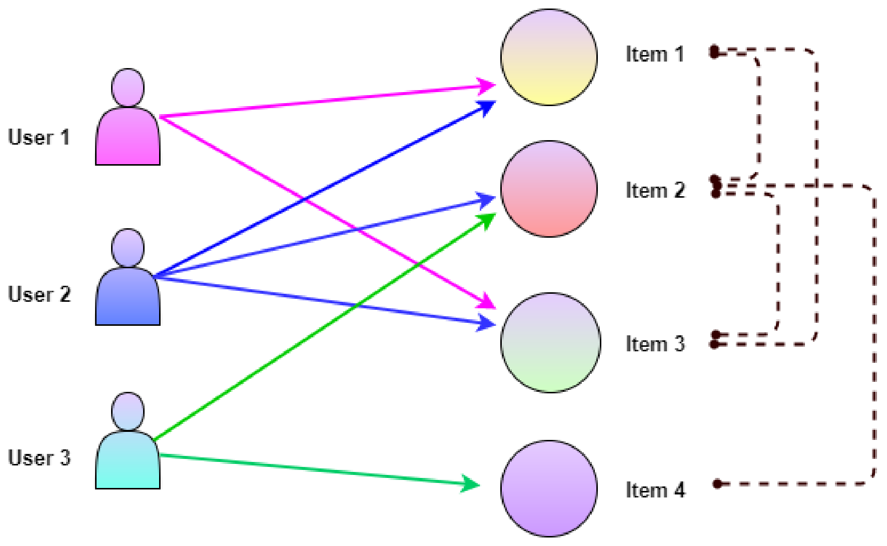


Figure 2.5: Example of item based Collaborative Filtering. *In this example unary preferences are given, which are represented by solid lines between the figures(users) and the circles(items). Item-item interaction is shown by dotted lines. Recommendation for user 3 can only be done using item 1 and item 3.*

Collaborative filtering algorithms can be further divided into memory and model based algorithms. An important subclass of the latter is the latent factor model (LFM). LFM is a generalized model. It consists of different models like single value decomposition, matrix factorization, factorization machines, etc. Each of their applicability varies with the nature of the data being applied on. Collaborative filtering algorithms, usually, give better results than content-based filtering if the user-item matrix is dense. However, in certain situations CF algorithms may also give good results in case of sparsity, but in most cases, content-based filtering algorithms give more accurate results when sufficient interaction data (or events) is available.

2.2.4 Pros and Cons

There are several advantages and drawbacks in Content Based Filtering recommender systems when compared to Collaborative Filtering recommender systems [36]. They are summarised in Table 2.2

	Content Based Filtering	**Collaborative Filtering**
Domain	Analyisis of domain is required. This allows to create accurate models	Not required. It works for any kind of data.
User Knowledge	Takes account of each user's independent preferences. Allows for personalised recommendations	Does not take into account user's individual preferences.
New Items	can analyse any new item due to presence of item model and domain knowledge	has new item problem
Flexibility	Not Flexible. Model has to be changed for any change in user or item	It is flexible
New User	Cold start problem exists	Cold start problem exists
Sparsity	Need to have sufficient interaction data	Performs better on sparse data
Overspecialisation	Has overspecialisation problem	Popularity bias

Table 2.2: Pros and Cons of Content Based Filtering when compared with Collaborative Filtering

2.3 Factorization Machine

In 2010, Steffen Rendle, introduced a seminal paper in the world of machine learning. In this work, Rendle described a concept known as a factorization machine. Factorization machines can be compared to support vector machines (SVMs) with a polynomial kernel. SVMs are widely used as general predictors. However, SVMs have known weaknesses, some of which are addressed by Rendle's factorization machines. SVMs function best on dense data. In SVMs, the input variables are still independent variables even though the polynomial kernel attempts to model the interaction among the variables. Factorization machines were designed to address these weaknesses. Firstly, no training examples are required in the model parameters, making the models much more compact. Factorization machines perform extremely well on sparse data, including data of very high sparsity. Additionally, through proper feature engineering, the machines can mimic the best specialized factorization models developed for very specific situations. This allows them to be applied in a multitude of situations where a

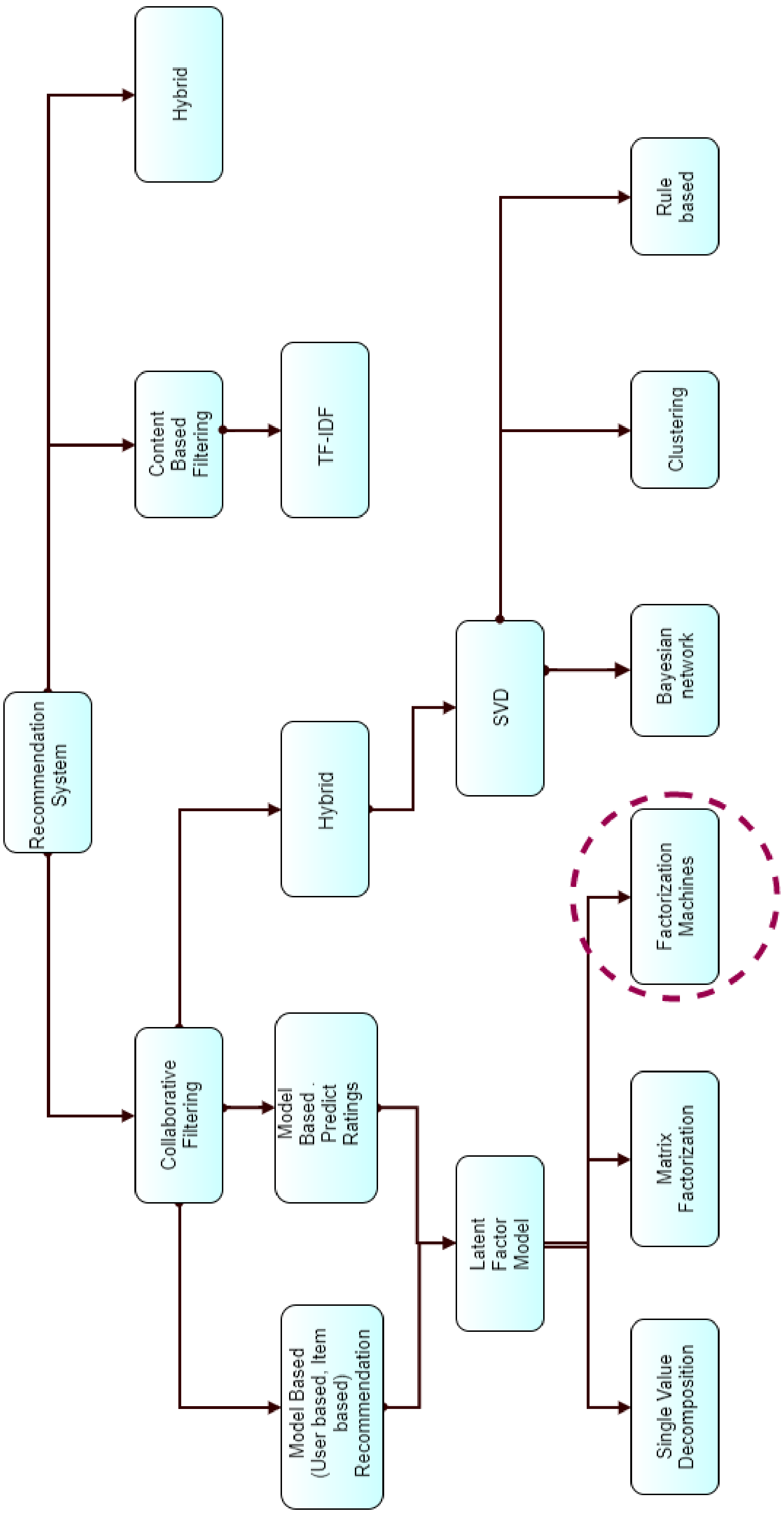


Figure 2.6: Placing of Factorization Machine amongst other recommendation system models. *We can see that FM fall under CF, but FMs are quite powerful in the sense that they can mimic a number of other models by feature engineering. FMs are best suited for our requirements as it incorporates pairwise interaction of user, item and their auxiliary features.*

specific form of learning algorithm and predictor is required. Figure 2.6 shows the placing of FMs amongst other recommendation systems.

2.4 Introduction to Expertise

Expertise discovery is fundamental to recommending the best responders. Understanding the nature of experts, their activity behavior, and their role is important for this study, The classification of expertise as described in Collins and Evans' periodic table of expertise[14] includes ubiquitous expertise (knowledge that comes from primary literature) and specialist expertise (knowledge that comes from the process of experimentation). In our work, apart from these two classifications, we also use a third category of expertise, called social expertise. We discuss more about the types of expertise and how to measure them in chapter 4

In the context of question answer system, social judgment is critical for expert identification. Apart from answering questions, asking a question and posting a comment may also provide evidence of user expertise. However, since answering a questions directly reflects the knowledge of a user, We limit our work for expertise characterization within the scope of answers.

2.5 Expertise Characterization

Compared to the previous problem of retrieving relevant questions and answers for a new question, there are fewer works aiming to solve the problem of finding the best answerers for a new question. The task of expert recommendation is predicting the best users who can answer a newly posted question. A ranked list of best answerers can be returned based on the similarity between the query and users history. To locate the users with desired expertise, quantitative measures of expertise are defined in [37]. They describe how to obtain these measures from a software project's change management system. They also presented evidence to validate this quantification as a measure of expertise.

Two general strategies for expert searching given a document collection are presented in [7] by using generative probabilistic models. Experts are found by mining expertise from email communications in [15]. Profile-based models for expert finding on general documents are proposed in [18]. There is also some research in question answerer recommendation. A new topic model which can simultaneously discover topic distribution for words, categories and users in a QA community is introduced to find a ranked list of answer providers [19]. Latent Dirichlet Allocation model is used in [34] and it has been combined with user activity and authority information to find the best answerers. [33] models expertise according to content similarity between the answering questions of users and test questions. If the test question is similar to the questions answered by the user in the past, he/her would have more probability and ability to answer this question. They use content-based model and peer-expertise model to model the strength between users and questions. Some methods of measuring expertise used in above mentioned works are described in the following sections.

2.5.1 Z-Score

The $Z_{score} = \frac{a-q}{a+q}$ is an often used metric for expertise which measures expertise according to the number of posted questions q and the number of posted answers a [51].

This measure tries to capture the combined answer and asking pattern and how different it is from a user that asks and answers questions with a probability $p = 0.5$. If a user answers more questions than he/she asks, the Z_{score} is positive, if the number of questions and answers is equal, the score is about 0 and if a user asks significantly more questions than answers, the Z_{score} is negative.

2.5.2 Reputation

Reputation or User score is the experience score that the Community Question and Answering system provides for a given user. For instance on Stack Overflow the user score is called reputation, while on Yahoo! Answers the user score are referred to as points.

User score is used as an expertise measure in some researches [38, 30, 20]. User score is used because it is readily available and intuitively represents the expertise of the user. Users of the platform have only one way of knowing if an answerer is an expert, which is looking at the answerers score.

However there are some notable disadvantages of user score as an expertise measure. First of all, the user score is not topic dependent. A user that has gathered his score on topic t does not have to be an expert at any topic $t' \neq t$. Second, it has been shown that user score can be closely related to the number of answers a user has provided [50, 20]. It has also been suggested that if a user is highly active it does not mean that the user is an expert [50], although high active users and expert users can overlap.

Both Z score and Reputation methods suffer due to their being independent of a topic and high reliance on user activeness, thus favouring highly engaged users over the ones that provide high level contributions.

2.5.3 MEC

Mean Expertise Contribution (MEC) is a measure that is created to specifically reduce the importance of the level of activity of the user in the expertise judgment [50]. In the process this approach also incorporates topic dependency for the level of expertise. Thus, this measure solves both problems that exist in the most popular measures of expertise. MEC or mean expertise contribution values three expertise factors like -

1. Answering Quality,
2. Question Debatableness, and
3. User Activeness

For a topic t, MEC is calculated as,

$$MEC_{u,t} = \frac{1}{|Q_t^u|} \sum_{\forall q_i \in Q_{u,t}} \mathcal{AU}(u, q_i) * \frac{\mathcal{D}(q_i)}{\mathcal{D}_t^{avg}} \tag{2.3}$$

where,

- $\mathcal{AU}(u,q_i)$ is the utility of the answer provided by the user u to question q_i. It is calculated as $\mathcal{AU}(u,q_i) = \frac{1}{Rank(a_{q_i})}$, i.e. the inverse of the rank of the answer provided by u for question q. The larger $\mathcal{AU}$, the higher the expertise level shown by the users in question q_i

- $\mathcal{D}$ is the debatableness of the question q_i, calculated as the number of answers $|A_{q_i,t}|$ provided for question q_i

- $\mathcal{D}_t^{avg}$ is the average debatableness of all the questions related to the topic t, calculated as $\frac{1}{Q_t} * \sum_{\forall q_i \in Q_t} |A_{q_j,t}|$

The value of MEC varies between 0 and 1 where MEC=1 indicated that the user u, on average, provides the best answer to averagely debatable questions. MEC=0.5 shows that u ranks second in answering averagely debated questions, or ranks first in answering less debated questions.

2.6 Methods

Some methods that will be later used in our work are introduced in this section.

One method of judging the quality of a particular model is by residuals. That means the model is fit using all the data points and the prediction for each data point is compared with its actual output. The mean absolute residual error is calculated. However, **Cross validation** is a model evaluation method that is better than residuals. The problem with residual evaluations is that they do not give an indication of how well the learner will do when it is asked to make new predictions for data it has not already seen. One way to overcome this problem is to not use the entire data set when training a learner. Some of the data is removed before training begins. Then when training is done, the data that was removed can be used to test the performance of the learned model on "new" data. The holdout method is the simplest kind of cross validation. The data set is separated into two sets, called the training set and the testing set. The function approximator fits a function using the training set only. Then the function approximator is asked to predict the output values for the data in the testing set (it has never seen these output values before). K-fold cross validation is one way to improve over the holdout method. The data set is divided into k subsets, and the holdout method is repeated k times. Each time, one of the k subsets is used as the test set and the other $k-1$ subsets are put together to form a training set. Then the average error across all k trials is computed. The advantage of this method is that it matters less how the data gets divided. Every data point gets to be in a test set exactly once, and gets to be in a training set $k-1$ times. The variance of the resulting estimate is reduced as k is increased. The disadvantage of this method is that the training algorithm has to be rerun from scratch k times, which means it takes k times as much computation to make an evaluation. Also in cases where the sequence of data is a factor in itself, k-fold cross validation is a poor choice.

Grid Search is method of systematically working through multiple combinations of parameter tunes (hyperparameter tuning), cross validate each (holdout or k-fold)

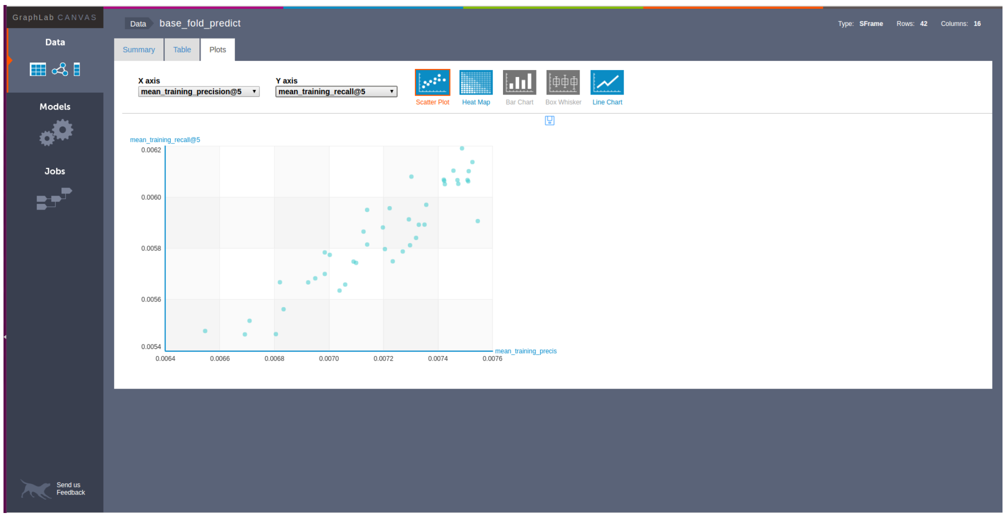


Figure 2.7: Screenshot of Graphlab Canvas Interface. *We can plot graphs, see distribution of data, compare models, and manage jobs using this interface.*

and determine which one gives the best performance. Using grid search, we can work through many combination only changing parameters by a bit.

2.7 Tool : GraphLab

For our experiments, we chose to use the **GraphLab** tool. GraphLab is a machine learning tool that can handle large data sets which results into scalable machine learning. It is an open source project and has been designed considering the scale, variety and complexity of real world data. Figure 2.7 shows a screenshot of graphlab canvas.It has state of the art machine learning algorithms including deep learning, boosted trees, and factorization machines and is user friendly. We required a tool that can handle millions of data points, and visualize it easily. With the presence of SFrames, graphlab could analyze terabyte scale data at interactive speeds. Thus, Graphlab was a good choice for us to go forward with.

2.8 Conclusion of Chapter

In this chapter we look at previous research work done in the field of community question answers. The problems addressed in this field are also discussed. We then look into Question Answer Routing, and ways adopted to improve it. We then introduce recommender system as a basis for question routing. Different types of recommender systems are studied in detail. We then introduce factorization machine, and discuss why it is best suited for our work. In the end we introduce expertise and some ways present in literature to measure. Few methods and tools that will be used later in our study are introduced in the last section. Research work done on expertise measurement limits to the question answering history and expertise based on a users' presence on the question routing platform. Studies like [49, 14, 8, 48] prove that in order to estimate a person's actual knowledge, we need to go beyond question routing platforms and take the advantage of a users' cross-platform domain specific behavior and advocate to capture different forms of expertise and expert behaviour. In our work, we target this approach and harness the knowledge of a user using his cross-platform behaviour.

Chapter 3

Conceptual Framework and User Modeling

In order to measure the effect of expertise for question routing using data from multiple web platforms, in this chapter, we first present and discuss the overview of flow of process adopted. Then we will discuss the data collection pipeline to create a user base to work upon. In the end of the chapter, we present methods of user matching used to create a multi-platform data user profile. This will be an answer to our RQ2.

3.1 Overview of Flow

The conceptual framework of my thesis work comprises of three aspects. Firstly, to understand the nuances of expertise and determining the measurement metrics of dif-

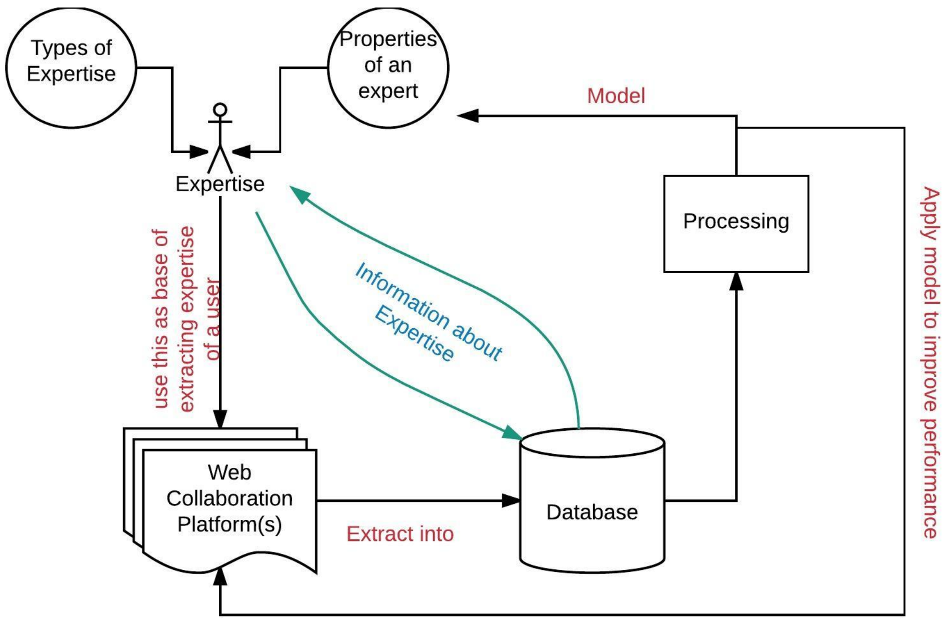

Types of Expertise
Properties of an expert
Model
Expertise
Processing
use this as base of extracting expertise of a user
Information about Expertise
Apply model to improve performance
Web Collaboration Platform(s)
Extract into
Database

Figure 3.1: Conceptual Framework

ferent types of expertise. Secondly, to gather data from multiple web collaborative platforms and to create a user profile out of it. Lastly, to apply the expertise metrics on the user profiles in order to provide better rank lists of user when searching for potential answerers of a particular question.

The process of expertise characterization for question routing is a multi-step process and is summarized in Figure 3.1.

From figure 3.1, first, expertise is studied in general. Types of expertise are identified and then properties of each type of expertise are listed. Data from web collaborative platforms is collected and stored in a database. The information from this database is used to fine tune the properties of each type of expertise. The identified properties are then used calculate and extract expertise of each user in the database. This calculated expertise is also stored in the database.

The data from the database is fed into the recommendation system generating function. A recommendation model is created. This model is then used to predict and recommend expert users for an inputted question. The performance of the model is evaluated and accordingly the model is improved until we find a model that significantly improves the expert recommendation for question routing.

3.2 Data Collection Pipeline

User's data from web collaborative platforms like Stack-overflow, Github and Twitter is used in my work. Software developers use these three platforms [49] to organize code, seek and provide support and share their work. For example, a software developer shares his work with the community on Github. He organizes his code on Github. At the same time, he is active on Twitter and tweets about latest and interesting things. Whenever he is stuck during his work, he takes help from the open Stack-Overflow community. He posts questions and seeks answers from others users. Also, he writes regular answers on topics he has knowledge of. In this way the he is present on every platform and leaves traces of his knowledge engulfed within his actions. We can take advantage of his cross-platform presence. An expert in the field of software development could be modelled using data from these three platforms. Thus, my first task was to find the common set of users with high accuracy levels of match between these platforms.

In order to create user model consisting of common users from Twitter, Stack-overflow and Github, We decided to divide our work into three parts-

- Find Common users between stack-overflow and github, (part 1)
- Find Common users between github and twitter, (part 2)
- Find Common users between stack-overflow and twitter. (part 3)

This division of work was necessary to extract the final set of common users between all these 3 platforms (part 4), as one can see in the Figure 3.2. For measuring various expertise of a user (especially social expertise), all parts were required and not just part 4.

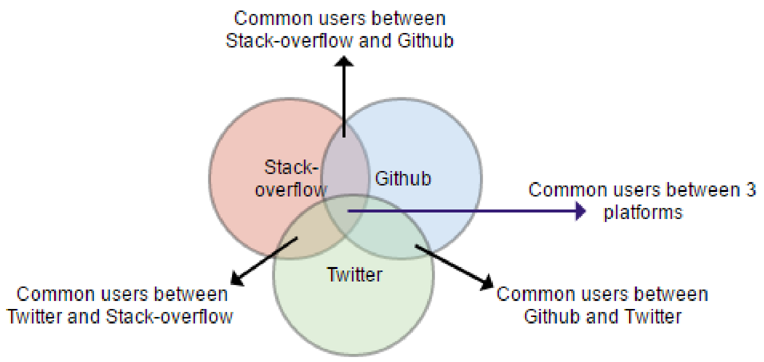

Common users between Stack-overflow and Github
Stack-overflow
Github
Common users between 3 platforms
Twitter
Common users between Twitter and Stack-overflow
Common users between Github and Twitter

Figure 3.2: Venn Diagram to show User Matching between three platforms

Stack-overflow being the question answering platform under study, we chose to start our work with existing measures of expertise characterization, i.e. based on previous answering history. Then to capture the cross-platform behaviour, it was best to first involve github and then twitter in the end. One could arguably say that why not twitter before or github before. Taking Twitter before does not make any sense as we want to study cross platform expertise for question routing, question routing is the central idea and not some other generic social behavior. Using github before also does not make that much of a sense as users on Github will be all experts (as per our measures) in their filed and only showcase knowledge of the topic they know about, whereas in Stack-overflow, we get a sense of a users' expertise levels for varied topics.

3.3 Collecting the data

Stack-overflow

The stack exchange meta data website[1] provides data dumps of stack-overflow. We used the September, 2016 data dump. This means we had the question, answer, comments and other related details of all stack-overflow users till September 2016. However, this data-set did not have email-id of users registered after 2014. Due to changes in privacy policy, we could not get that data. All the data was extracted into tables for further usage.

Github

The github data till November 2016, was downloaded from the regular dumps provided on their website[2]. This data contained all information about users, their projects, repositories, comments, etc. The email ids of users was included till May, 2016. The emails of users registered between May 2016 and November 2016 was collected explicitly from the managers of github data.

[1]https://archive.org/details/stackexchange
[2]http://ghtorrent.org/downloads.html

Twitter

Data from twitter was collected only for existing and matched users of stack-overflow and github. Rest API's of twitter were used to crawl user's data. Also, twitter's search page[3] was scrapped to get maximum data. The process of data collection is discussed further in the chapter.

Stack-overflow	
Number of questions	12,350,000
Number of Answers	19,974,952
Number of users	5,987,285
Github	
Number of users	14,380,097
Number of projects	39,635,052
Number of commits	66,819,469

Table 3.1: Dataset Statistics

Table 3.1 summarizes the important statistics of my dataset.

3.4 User Matching

Once the data has been collected, the next step is to link users between the three platforms. This is also our first step in the direction to answer second research question (**RQ2**). We adopted the matching strategies for direct and attribute matching introduced in [43]. We extended fuzzy matching adopted in [43] to match user accounts between Stack-Overflow and Github.

- **Direct Matching** : Direct Matching or explicit matching refers to the matching of user profiles when explicit information about other platform's profile is given in data. Explicit links are provided by users in one platform to the their accounts in other platforms.

  Starting from our built dumps of Stack-Overflow and Github, we perform explicit matching by analyzing user-provided links from the user profiles in each of these platforms to the other platforms. We consider this a very reliable method for account linking because matching information are provided by users themselves, with strong incentives for truthful linking.

  From Stack-Overflow to Github, Twitter. We analyze Stack-Overflow user profiles to find explicit links to Github and Twitter users. For Stack-Overflow users that provide links to their Github link, we parse the direct links, which are in the form of https://github.com/GitHubLoginName and obtain their Github login names, i.e., GithubLoginName. For Stack-Overflow users that provide direct links to Twitter, which is usually in the form of http://www.twitter.com/Twitter ScreenName, we parse the Twitter screen name, i.e., TwitterScreenName. Both Github login name and Twitter screen name uniquely identifies one user in

[3]https://twitter.com/search-home

Github and Twitter, respectively. From Github to Stack-Overflow, Twitter. We analyze Github user profiles similarly to match user profiles in Stack-Overflow and Twitter.

The result of explicit matching is reported in Table 3.2. As it can be seen, we were able to match thousands of users between the three platforms.

Direct Matching		
With	**To**	**Number of Users**
Stack-overflow	Github	5659
Github	Stack-overflow	678
Stack-overflow	Twitter	5426
Github	Twitter	6260

Table 3.2: Explicit Matching

- **Attribute Matching** : Attribute Matching refers to the matching of user profiles using unique attributes of users' accounts to connect profiles across multiple platforms from the same user. Excluding directly matched users, attribute matching was performed.

  Stackoverflow and Github provide users with the option of registering their emails, which are encrypted into MD5 hashes in the data dumps. This technique is known from literature [47, 6, 43] to be a reliable way to match users by their email reference. There are in total 2,185,162 (≈36.5%) Stackoverflow users and 1,026,676 (≈7.1%) Github users with email hash. Email hashes were previously considered for matching users between Stackoverflow and Github in [47, 43].

  Besides using the email hashes explicitly provided by users, we exploited Gravatar [44] to increase the number of available hashes in both platforms. Gravatar-id is a unique image associated with an email hash. We find that many users use Gravatar to have a unique profile image across Stackoverflow and Github. By making HTTP request for a Gravatar profile image, we obtain a user's MD5 email hash [45]. We identified 4,752,335 (≈33.1%) Github users, and 5,902,395 (≈9.9%) Stackoverflow users with Gravatar email hash available.

$$\begin{aligned} query = &((StackOverflowUsers[emailhash] \cap GithubUsers[emailhash]) \\ &\cup (StackOverflowUsers[gravatarid] \cap GithubUsers[gravatarid]) \\ &\cup (StackOverflowUsers[emailhash] \cap GithubUsers[gravatarid]) \\ &\cup (StackOverflowUsers[gravatarid] \cap GithubUsers[emailhash])) \end{aligned} \tag{3.1}$$

  Combing email hashes explicitly provided by users, and implicitly revealed from their Gravatar Id, we use Query 1 for Stackoverflow-Github user matching, which encodes all meaningful joins between MD5 email hash and Gravatar Id attributes across the two platforms. The result of attribute-based matching is shown in Table 3.3. We finally obtained more than 767k exactly matched users between Stackoverflow and Github.

Attribute Matching		
With	**To**	**Number of Users**
Stack-overflow emailhash	Github emailhash	111,083
Stack-overflow emailhash	Github gravatarid	649,875
Stack-overflow gravatarid	Github emailhash	1,463
Stack-overflow gravatarid	Github gravatarid	5,489
Union		767,910

Table 3.3: Attribute Matching

- **Fuzzy Matching** : Fuzzy Matching refers to the matching of user profiles using implicit data provided in the data. Excluding directly and attribute base matched users, fuzzy matching was performed. This also included all users who did not have any email hash or gravatar-id.

  - **Stack overflow - Github**

    Stack-overflow and github datasets had the *loginname* or the display name of the user as a common and mandatory field. We therefore chose to apply fuzzy matching on the login names of users. *Loginnames* being strings opened doors to apply various matching conditions that lead to fuzzy matching.

    In order to be sure that the matched user is actually the same user on both platforms, it was important to derive the complexity of *loginname* strings. More complex the string, more chances of the matched users being the same person.

    The following assumptions were made:

    * If a string comprises of special character such as '$@#^&%<>?{}!', then it belongs to the highest level of complexity
    * If a string comprises of both alphabets and numbers, but not special characters, then it is complexity of second degree.
    * If a string just has alphabets, then it may or may not be complex

    We applied the above strategy and matched the *loginnames*. If the match belonged to the first two criterion mentioned above (ignoring the case), then the match was considered as high confidence match. If the match belonged to the third category (not ignoring the case), then we compared it further based on the length of the string. If the length of the string was greater than equal to 8 (a threshold value chosen after experimentation), we considered it as a high confidence match.

    We manually checked profiles of 100 users, and found an accuracy of 82%. One limitation of this strategy was that, it relied on a *loginname* being matched, and then see if the *loginname* is complex enough to include as a high confidence matched user or not. We eventually decided to use this strategy because when we consider complexity of a loginname, we were not directly matching the user, but selecting the ones that have a high chance of belonging to the same user (in different platforms).

In the end we unioned all users matched by fuzzy matching. 41,935 users were matched.

– **Github, Stackoverflow - Twitter**

  Matching accounts from Stackoverflow and Github with Twitter accounts is inherently more difficult, since Twitter profiles need to be obtained via Twitter API services.

  All users profiles that had an additional information in the form of either a web url or profile image were considered for fuzzy matching. This selection of limiting the users to only ones having an extra information was based on the fact that at a later stage, only the users with a matched account based on the below mentioned criterion will be considered as high confidence match, thus the users not having that information would be eventually excluded. Therefore, we did not take those users into consideration, and tried to reduce the complexity of the task.

  To move forward with user matching, here *loginnames* was chosen as a basis of matching twitter, stackoverflow and github accounts. According to Twitter APIs, matching could either be done based on *id*, *screen-name* or *name*. *Id* of a user on github, stackoverflow and twitter is different. *Names* are more complex to categorise and cannot be used as primary input, as it could lead to false positives. Lastly, *loginnames* were the only feasible option to send as an input and match with the *screenname* of twitter.

  To move forward, we decided to do categorization of *loginnames*. This was done to understand the effects of various *loginnames* on twitter. Twitter not being case-sensitive, we categorized the loginnames in the following categories

  ∗ Alphabets (only alphabets)
  ∗ Numeric (only digits)
  ∗ Alphanumeric (both alphabets and digits)
  ∗ Special (having any special character excluding'-')
  ∗ Dashed (having a dash ('-'), implying a conjugation of more than one work)
  ∗ Space (having a space in between words. This type of *loginname* was only found in stackoverflow data)

  Dashed and Space were chosen as two separate categories, because in Twitter, a search with a dash(-) is considered as one whole input, whereas a search with a space is considered as two separate word (not necessarily occurring together).

  A distribution of *loginname* belonging to a certain category was found for inputted github and stackoverflow users. This distribution can be seen in Table 3.4. The most stark information in this table is the absence of loginname with special characters in github dataset. On further investigation, we found that github user-names does not allow[17] any special characters except '-,_,'.

Distribution of *loginname* categories				
Category	**Github**	**StackOverflow**	**%github**	**%stackoverflow**
Alphabets	3,197,470	872,517	0.7316	0.2625
Numeric	9,194	1,034	0.0021	0.0003
Alphanumeric	886,184	1,354,009	0.2028	0.4074
Special	0	40,315	0	0.0121
Dashed	277,728	16,667	0.0635	0.0050
Space	N.A.	1,039,153	N.A.	0.3127
Total	4,370,576	3,323,695		

Table 3.4: Distribution of *loginname* categories for Github and StackOverflow

Lookup and Search

Two types of query requests were here considered, namely Twitter REST API and Twitter.com Search. We refer to them as *Lookup* and *Search* [43] respectively. The Lookup method returns the full profile information of the user corresponding to a given screen name, id or name. Using Twitter REST API, rate limit of 100 requests per 15 min window has to be followed. On the other hand, Twitter.com Search permits to process only one input per request. Although, twitter search being less efficient, it is more flexible in terms of the input. It can accept any textual input. There is another searching method on Twitter, Search API. Using search API was not a good option as it did not give access to historical data on Twitter.

Based on the data we had for github and stackoverflow, we considered the following options for input -

* Login names, and names of users' Stackoverflow and Github accounts,
* Website urls and profile image of users' Stackoverflow and Github accounts,

Website urls, (leaving facebook and instagram urls, as they could lead to false positives) if matched indicate both the accounts belong to the same person. Here we made an assumption that all the urls provided, belonged to the user himself. Thus if a website url was matched, then the accounts on matched platforms, definitely belonged to the same user.

Accuracy of *Lookup* and *Search* methods

To assess the performance of *Lookup* and *Search* methods, we took a random sample dataset of 2000 users each from github and stack-overflow. Figure 3.3a shows the distribution of stackoverflow and github loginnames according to various categories. This distribution is similar to the distribution of loginnames of whole dataset (chi-square(4) is 1.45244 for github and chi-square(5) is 6.53691 for stack-overflow) shown in Table 3.4. We can see from Figure 3.3a the majority of loginnames are only alphabets, followed by alphanumeric.

To understand how different categories behave when inputed in Lookup and Search, we in Figure 3.3b analyse the percentage of Github and Stackoverflow login names that have at least one candidate returned by *Lookup* and *Search*. High values indicate higher probability that the user can be matched. We can see that the *Lookup* performs better for numeric ("Numeric") input and *Search* for all other categories of input.

For each category, we manually checked the matched accounts. A user was considered to be matched with a Twitter account if there was explicit Twitter information (e.g., personal website, profile image, profile description) that can identify the user with high confidence. Table 3.3c shows that *Search* performs better than *Lookup*, especially for login names that belong to the "space", "special" and "dashed" categories. The least gain of *Search* over *Lookup* is in the category "Number" (negative). Considering Figure 3.3b and the higher efficiency of *Lookup* method, we chose to use *Lookup* for login names in the "Number" category, and *Search* for the other categories.

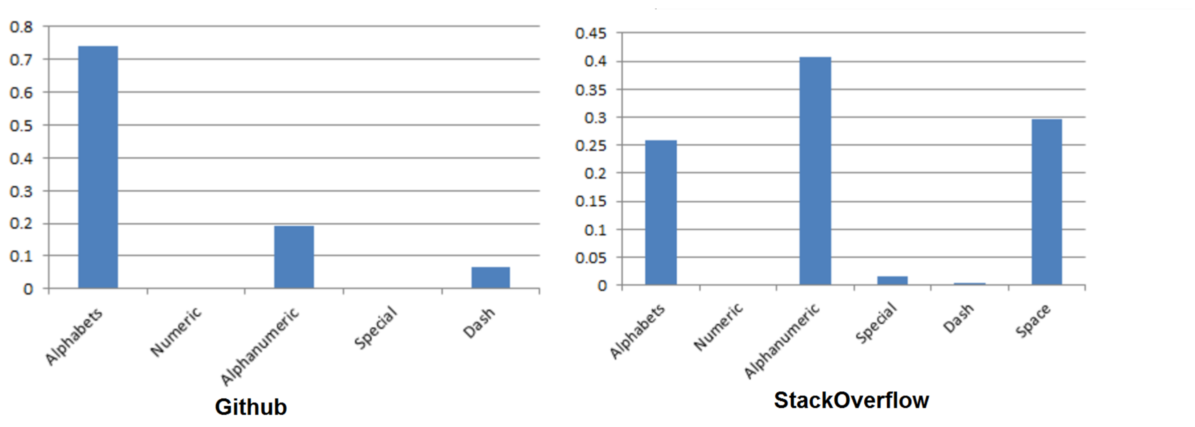
Workflow of Fuzzy Matching

Figure 3.4 shows the work flow of fuzzy matching using *Lookup* and *Search* methods. Given a users' login name, it first determines whether to use *Lookup* or *Search* (based on evaluation done is previous section), then it checks Twitter profiles for high confidence account matching.

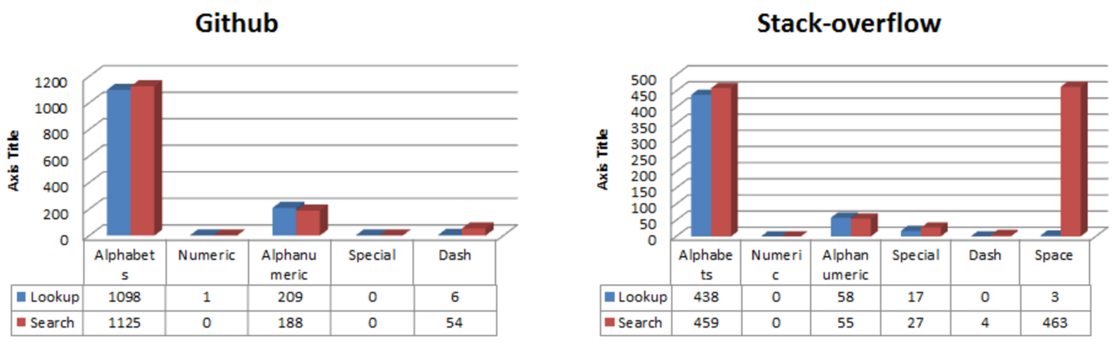
A user is matched to a Twitter account if he/she meets the following criteria:

∗ the website attribute of the user's Twitter profile is exactly the same as the website of his/her Stackoverflow or Github profile
∗ the twitter profile picture needs to be highly similar to her/his profile picture in Stackoverflow or Github profile
∗ the length of login name is above the threshold value

If the above criterion are matched, then we considered the match as a high confidence match. If it was a low confidence match (i.e. either of the three conditions mentioned above were not matched) in Search method, we performed an additional step. We inputted some mutations into the input login name string and then again performed website and profile image match. If after this step, a high confidence match was not returned, then we inputted the name of stackoverflow or github user (instead of his login name). We again performed the above mentioned steps. If we found a match, then it was taken as a high confidence match otherwise no match. For all high confidence matches, all relevant information was scrapped and collected from Twitter in our database.



(a) Login name distribution for sample dataset of 2000 users



	Alphabets	Numeric	Alphanumeric	Special	Dash
🟦 Lookup	1098	1	209	0	6
🟥 Search	1125	0	188	0	54

	Alphabets	Numeric	Alphanumeric	Special	Dash	Space
🟦 Lookup	438	0	58	17	0	3
🟥 Search	459	0	55	27	4	463

(b) Number of candidates (>=1) returned by *Lookup* and *Search* for each category in sample dataset of 2000 users

Accuracy of twitter user matching for different login name categories			
Category	**Lookup**	**Search**	**Gain**
Alphabets	0.8	2	+1.2
Numeric	7.6	0	-7.6
Alphanumeric	0.17	0.5	+0.33
Special	0	5.8	+5.8
Dashed	0	5.07	+5.07
Space	0	5.9	+5.9

(c) Accuracy of twitter user matching for different login name categories in sample dataset of 2000 users

Figure 3.3: Accuracy of Twitter user matching using *Lookup* and *Search* for different categories of Github and Stackoverflow login names.

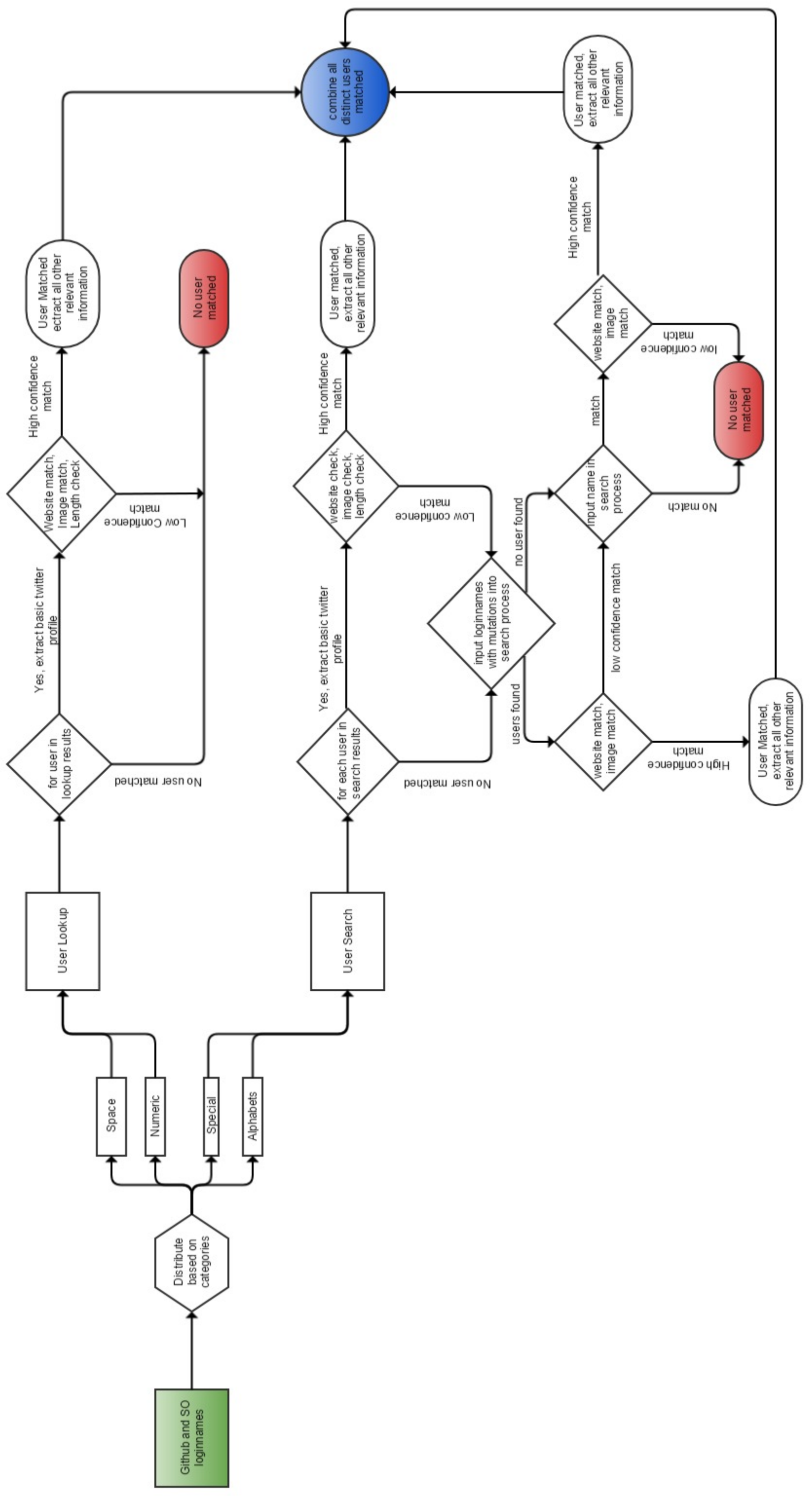

Github and SO loginnames
Distribute based on categories
Space
Numeric
Special
Alphabets
User Lookup
User Search
for user in lookup results
No user matched
Yes, extract basic twitter profile
Website match, Image match, Length check
Low Confidence match
No user matched
High confidence match
User Matched ectract all other relevant information
for each user in search results
No user matched
Yes, extract basic twitter profile
website check, image check, length check
Low confidence match
High confidence match
User matched, extract all other relevant information
combine all distinct users matched
input loginnames with mutations into search process
no user found
users found
input name in search process
No match
match
website match, image match
low confidence match
No user matched
High confidence match
User matched, extract all other relevant information
website match, image match
low confidence match
High confidence match
User Matched, extract all other relevant information

Figure 3.4: Work Flow of Fuzzy Matching

The websites matched in criterion 1 ignored ambiguous websites such as http://facebook.com, http://instagram.com, which could bring false positives for the matching. The process of profile image similarity check in criteria 2, was performed using image hashing algorithm called *'Perceptual Hash Algorithm'* [23]. This algorithm creates a type of fingerprint (hash) for each image and compares the two fingerprints. If the fingerprints match, or are close, then it has high chance of being same. This algorithm returned a percentage of similarity between the two inputted images.

Using profile image, as a stand alone comparison basis, did not prove to be the best choice. When we manually checked our experimental 2000 users, this algorithm returned a high value of similarity even for default images, and images that had similar color patterns but were not same. So, perceptual hash algorithm using a stand alone condition was not a good choice. In order to come up with a more generalized condition for profile image matching, we inspected the inputted *loginnames*. On manually checking the profile images, we found that the users having greater lengths of input and high profile image similarity percentage, were a high confidence match. Thus, on experimenting with various combinations of length and similarity levels, we came up with an optimum combination.

∗ If length is greater than 9 and similarity is between 80 and 90, or if length is greater than 7 and similarity between 90 and 98, or if similarity is greater than 98, then a match is a high confidence match. This combination performed fairly for alphabets, alphanumeric and others.

∗ For dash (had larger lengths in general because of more than one word), the optimum combination was, if length is greater than 11 and similarity is between 70 and 90, or if length is greater than 9 and similarity between 90 and 98, or if similarity is greater than 98.

∗ Default profile images were excluded in the above study.

Table 3.5 shows the user matching results. We analyzed 7,694K stackoverflow and github accounts, specifically 10K by *Lookup* and 7684K by *Search*. The number of accurate matched users are 16 and 63K respectively, with a total of 63,176 user accounts matched on Twitter.

Twitter User Matching Results			
Search Method	**User Analyzed**	**User Matched**	**Matching %**
Lookup	10,228	16	0.156
Search	7,684,043	63,160	0.82
Total	7,694,271	63,176	0.82

Table 3.5: Results for Twitter User Matching with Github and Stackoverflow users

We analyzed 5,987,285 Stackoverflow users and 14,380,097 Github users in fuzzy matching. 41K users were matched between Stackoverflow and Github. 38K were matched between Github and twitter. 24K were matched between Stackoverflow and twitter.

3.5 Conclusion of Chapter

In this chapter we first looked at conceptual framework of our work. We discussed the data requirements for solving our research questions. In the next section, we looked at the data collection pipeline and discussed the process of data collection in the subsequent section.

We address our RQ2 in the form of User Matching in section 3.4. User profiles of Stackoverflow, Github and Twitter were matched using direct, attribute and fuzzy matching. Direct matching was done to match user profiles where explicit information was provided. The number of users matched using direct matching is shown in table 3.2. Attribute matching was done using *gravatarid* and *emailhash* of a user across Stackoverflow and Github. We matched a total of 767,910 users across SO and GH using attribute matching. Both direct and attribute matching was adopted from [43] Fuzzy matching using implicit data from SO, GH and TW with the help of *Lookup* and *Search* methods was adopted from [43] to match users of SO and GH with the users of TW. We extended this fuzzy matching to match users between SO and GH. We categorize *loginnames* and apply *Lookup* method for *numeric loginnames* and *Search* method for other categories. We perform different stages of matching as summarised in figure 3.4. In the end, we combined the results of all three matching methods. We analyzed a total of 6 million stackoverflow user profiles and 14 million github profiles, and matched them across three platforms (SO-GH-TW). Table 3.6 summarizes the number of users matched.

Table 3.6: Number of Users Matched between Stack-overflow, Github and Twitter

Stack-overflow - Github	816,182
Stack-overflow - Twitter	29,710
Github - Twitter	45,136
Total Matched Users	891,028
Unique users having presence on all 3 platforms	**59,812**

From Table 3.6, we can see that 816K users were matched between Stackoverflow and Github, 45K between Github and Twitter, and 29K between Stackoverflow and Twitter. Total 891K users were matched, out of which 59K users belonged to all three platforms.

We learnt from the process of User Matching that, between Stackoverflow and Github, we have a higher percentage of common users than between So-TW or GH-TW. The users also tend to give more of their Github profiles in their Stackoverflow answers. Github users are more connected on Twitter as compared to Stackoverflow users. Although, we match almost 890K users across three platforms, but the number of users having presence in all three platforms is very less (6.7%). This number can be increased in future if we fine-tune our fuzzy matching technique to include more implicit data.

Chapter 4

Measuring Expertise

Expertise has been measured in a number of ways in previous research works. Some of the works are mentioned in chapter 2. Majority of these works measure expertise in accordance to user activeness on CQA service platforms. Very few works [49, 50] measure expertise in accordance to actual actionable knowledge of the user. Although, they are a major improvement from expertise based on user activeness, yet, they rely only on the previous answers provided by the user on question answering platforms. A working hypothesis is that, measuring expertise of a user as a function of users' profile built from multiple web collaborative platforms is more accurate measurement of user's actual knowledge. In the following three sections of this chapter we introduce the method that we adopt to answer the three research questions as introduced in the introduction chapter.

4.1 Types of Expertise

Expertise is a property of an individual, or a community of individuals, which affects the reliability and quality of performance [13] in a given domain of knowledge or practice [49]. Experts are perceived as those users who, given a question or a task, can provide appropriate answers or perform tasks timely and correctly. Sociologists have extensively studied the relationship between expertise and expert behavior in communities. Harry Collins and Robert Evans in their book Rethinking expertise [14] propose the *"Periodic Table of Expertise"* as an attempt to provide a conceptual framework for the organization for different types of expertise. In their classification, the tacit knowledge expressing domain-specific expertise can be of two main types, namely ubiquitous, i.e. knowledge that comes from primary literature (e.g. manuals, books, Web), and specialist, i.e. knowledge that comes from the process of imbibing in a discipline and that allows its holders to contribute to the domain to which the expertise pertains [49]. In our work, apart from these two classifications, we also introduce a third category of expertise called social expertise, i.e. knowledge gained with the help of social bonds.

- **Ubiquitous** : If a person seems to have technical knowledge but has not acquired it from experimentation and experience then its ubiquitous expertise. The definition for ubiquitous expertise in [14] is given as - *ubiquitous expertises are those, such as natural language-speaking, which every member of a society must* *possess in order to live in it; when one has a ubiquitous expertise one has, by definition, a huge body of tacit knowledge*. Ubiquitous tacit knowledge refers to the kind of knowledge that comes with reading primary or seemingly primary literature, i.e. books, manuals, guides, articles, papers, etc. The possession of ubiquitous knowledge gives an impression of technical mastery. However it is specialist expertise that actually refers to mastery.

- **Specialist** : If a person has acquired knowledge by self experimentation and experience then that knowledge is called specialist expertise. This is the highest form of expertise. Specialist knowledge can only be mastered through gradual acquisition, and it is the only type of knowledge that allows its holders to contribute to the domain to which the expertise pertains. People with specialist knowledge have the ability to actually do things.

- **Social** : Apart from ubiquitous and specialist expertise, from the perspective of world wide web, we have found another type of expertise, called social expertise. In current social society, people interact and share knowledge using online platforms like tumblr[1], Plurk[2], twitter[3], facebook[4], instagram[5], etc.. These platforms act as a social platform where people share their daily experiences, thoughts, knowledge, etc. with their friends. It has become an integral part of our everyday lives. If a person has acquired some technical knowledge about a certain topic by the help of a peer (who may or may not be a specialist) or by just talking or reading his knowledge/experiments, then this type of expertise is called social expertise. The knowledge acquired with the help of social bonds is called social expertise. Social platforms assist such behavior and showcase existence of such people. This is the lowest form of expertise amongst the three types.

4.2 Mapping user behaviour

In this section we show the mapping user behaviour with different types of expertise. First we describe about what user actions are possible on Twitter, Stack-Overflow and Github. Then we show the what actions relate to which type of expertise. In the end we show means to calculate each type of expertise.

4.2.1 User Actions

User performs a number of actions on web platforms. These actions result in the user performing all the tasks he needs to perform and get the desired results from the platforms. The users registered on Stack-Overflow, Twitter and Github also perform actions that are mentioned below but are not limited to the list. Users not registered on these platforms also perform limited number of actions but they are not under consideration.

[1]https://www.tumblr.com/
[2]https://www.plurk.com/portal/
[3]https://www.twitter.com/
[4]https://www.facebook.com/
[5]https://www.instagram.com/

- **Twitter :** Users registered on twitter can - Tweet any information; Receive, send and delete tweets; send, receive and delete a direct message from another user; Reply to a message or tweet; Re-tweet, Un-tweet any tweet; Mention a user; Follow or Un-follow an account(could be a users or of an organisation or anything); Block another user's account; Approve or Deny Follower Request from another user; Favorite or Un-favorite any tweet; Add hashtag, photo and URLs in tweets; edit personal profile. Twitter is a micro-blogging site which is very simple to use as broadcaster or receiver. The user joins with a free account and Twitter name. Then the user can send broadcasts daily, or even hourly. He can go to the 'What's Happening' box, type 140 characters or less, and click 'Tweet'. To receive Twitter feeds, the user simply finds someone interesting (celebrities included), and 'follow' them to subscribe to their tweet microblogs. Once a person becomes uninteresting to you, you simply 'un-follow' them. The user then chooses to read his/her daily Twitter feeds through any of various Twitter readers.

- **Stack-Overflow :** Users registered on stack-overflow can - Post a question, post an answer; Comment on an answer or question; Search for a question to answer; up-vote or down-vote an answer; Favorite or Un-favorite a question; read notifications; Edit a question or answer; Attach tags to a question; Merge profiles; edit personal profile;. Stack-Overflow is a question and answer site for professional and enthusiast programmers. This site is all about getting answers. It's not a discussion forum. Just questions and answers. Good answers are voted up and rise to the top. The best answers show up first so that they are always easy to find. The person who asked can mark one answer as "accepted". Accepting doesn't mean it's the best answer, it just means that it worked for the person who asked. All questions are tagged with their subject areas. Each can have up to 5 tags, since a question might be related to several subjects. The user earns badges and unlocks new privileges like the ability to vote, comment, and even edit other people's posts when his/her reputation score goes up.

- **Github :** Users registered on github can create a new repository; import an existing repository; create a new gist; create new organization; add members to an organization; write content; edit content; commit changes; create, merge and delete a branch; Follow other users; Watch an existing project; manage an organization; star a project; fork a repository; make pull requests; follow another user; approve or deny follow requests from other users; edit personal profile;. Github is a platform where developers store their projects and network with like minded people. Each project has its own repository which is a location where all the files for a particular project are stored. "Forking" is when you create a new project based off of another project that already exists. If a user finds a project on Github that he/she likes to contribute to, they can fork the repository, make the changes they'd like, and release the revised project as a new repository. The social networking aspect of Github allows projects to grow. Each user on Github has their own profile that acts like a resume of sorts, showing their past work and contributions to other projects via pull requests.

Figure 4.1 shows some important actions of a user that are of high importance in our study.

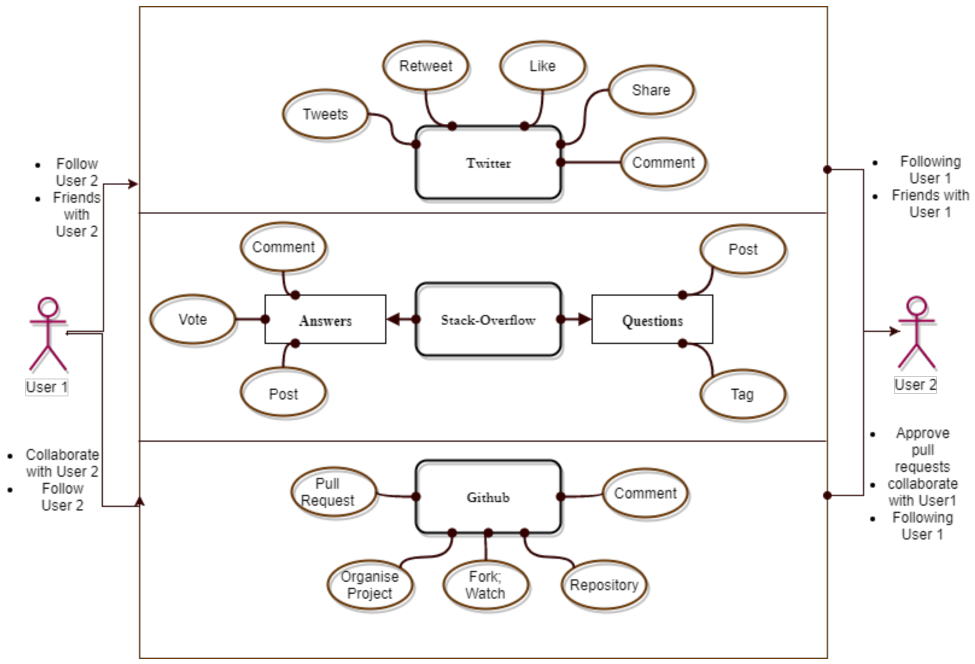


Figure 4.1: Diagram showing some user actions that a user performs on Twitter, Stack-Overflow and Github.

Their are also some additional derived entities related to a user on these platforms that are important for our study, they are :

1. **Score** : The actual number of votes received by an answer in stack-overflow. This is a summation of all up-votes and down-votes.
2. **Answer Reputation** : The reputation of an answer amongst all answers for a question on stack-overflow is calculated as,

   $$reputation = \frac{1}{rank} \tag{4.1}$$

   Rank is the normalized score of an answer belonging to a question. For a question, the answer with highest vote is ranked 1 and so on.
3. **Reputation Score** : The reputation of a user is the total score accumulated by a users' answers amongst all questions. This value is not just a summation of up-votes and down-votes, but it is also increased with gain in badges and other rewards.
4. **Topics** : Each question in SO is associated with a number or tags. Expertise of a user is always subjected to a particular topic. In our study, a topic is represented using tags. Tags from a question are extracted by tokenization, and then each tag is used as a topic.
5. **MEC** : Mean Expertise Contribution is calculated as per equation 2.3

4.2.2 User actions relating to types of expertise

Within the targeted platforms (Stackoverflow, Github and Twitter), we operationalise the concepts of ubiquitous, specialist and social knowledge in a set of user activities performed. In the scope of software development, actions are classified as specialist when they refer to actionable knowledge, i.e., actions or content that reflects evidence of practical competence. For example, source code shared on github as part of a project (own or someone else's) is a representation of specialist knowledge. Code snippets on platforms like Gist[6], Pastebin[7], or snipt[8] also comply to specialist knowledge. Specialist knowledge can also be seen within code snippets contained in answers and comments in Stackoverflow. This category also includes original tweets in Twitter that

- are related to software development topics and
- refer to actionable knowledge.

We consider questions on Stackoverflow also an example of specialist knowledge. Questions reflect an active attempt to acquire actionable knowledge. All other types of actions like tweeting on a topic, commenting, general answers, etc. are marked as ubiquitous knowledge. Frequent re-tweets of tweet written by a knowledgeable person, linking to works of their friends, etc. showcase social knowledge.

In our study, we measure expertise based on answers and comments written in Stack-overflow, tweets and re-tweets posted on Twitter and presence of a github account on SO and TW. This decision of limiting the number and types of user behaviour on these platforms was taken to first study the performance of types of expertise in question routing. The definition of types of expertise in terms of user behaviors associated with them can be increased in future work.

4.2.3 Measuring Ubiquitous expertise

To measure ubiquitous expertise, from the user modeled by matching data from SO-GH-TW, we consider the following aspects

- Explicit code snippets - the number of code snippets identified in all answers of a user belonging to a certain topic.
- Related Tweets - number of tweets having other users work related to a topic.
- Linked Tweets - the number of tweets having an explicit link to other users' github work for the topic.
- Linked answers - the number of other users github links identified in all answers of a user belonging to a certain topic.

[6]https://gist.github.com/
[7]http://pastebin.com
[8]https://snipt.net

UE or ubiquitous expertise was calculated by incorporating above three aspects. The formula for UE is:

$$UE_{u_i,t_i} = \frac{ExplicitCodeSnippet_{t_i} + RelatedTweet_{t_i} + LinkedTweet_{t_i} + LinkedAnswers_{t_i}}{2 * (\#TotalAnswers_{t_i} + \#TotalTweets_{t_i})} \tag{4.2}$$

where, $TotalAnswers_{t_i}$ is the total number of answers written by user u_i on topic t_i. $TotalTweets_{t_i}$ is the total number of tweets done by user u_i on topic t_i.

4.2.4 Measuring Specialist expertise

To measure specialist expertise, from the user modeled by matching data from SO-GH-TW, we consider the following aspects

- Explicit code snippets - the number of code snippets identified in all answers of a user belonging to a certain topic in SO.
- Explicit code comments - the number of code snippets identified in all comments of a user belonging to a certain topic in SO.
- Self Linked Tweets - the number of tweets having an explicit link to own github work for the topic
- Self Linked ReTweets - the number of retweets having an explicit link to own github work for the topic
- Self Linked answers - the number of links to own github work identified in all answers of a user belonging to a certain topic in SO.

SpE or Specialist expertise was calculated by incorporating above five aspects. The formula for SpE is:

$$SpE_{u_i,t_i} = \frac{\begin{matrix} ExplicitCodeSnippet_{t_i} + ExplicitCodeComments_{t_i} \\ + SelfLinkedTweet_{t_i} + SelfLinkedReTweet_{t_i} + SelfLinkedAnswers_{t_i} \end{matrix}}{2 * (\#TotalAnswers_{t_i} + \#TotalTweets_{t_i}) + \#TotalComments_{t_i}} \tag{4.3}$$

where, $TotalAnswers_{t_i}$ is the total number of answers written by user u_i on topic t_i in SO. $TotalTweets t_i$ is the total number of tweets done by user u_i on topic t_i in TW. $TotalComments t_i$ is the total number of comments written by user u_i on topic t_i in SO.

4.2.5 Measuring Social expertise

To measure social expertise, from the user modeled by matching data from SO-GH-TW, we consider the following aspects

- Linked Tweets - the number of tweets having an explicit link to work of other specialist users for the topic.
- Linked answers - the number of specialist users' github links identified in all answers of a user belonging to a certain topic.
- Friendship - the degree of friendship on twitter with specialist users.

SoE or Social expertise was calculated by incorporating above three aspects. The formula for SoE is:

$$SoE_{u_i,t_i} = \frac{LinkedTweet_{t_i} + LinkedAnswers_{t_i} + Friendship_{\forall specialist \in t_i}}{\#TotalAnswers_{t_i} + 2 * \#TotalTweets_{t_i}} \tag{4.4}$$

where, $TotalAnswers_{t_i}$ is the total number of answers written by user u_i on topic t_i in SO. $TotalTweetst_i$ is the total number of tweets done by user u_i on topic t_i in TW.

We conducted some statistical analysis to verify our expertise measurement techniques. The results are shown and discussed in next chapter.

4.3 Applying expertise measure to Question Routing

In this section, we apply the expertise measured in previous section to question routing. First we present model in factorization machines. Then we introduce hyper-parameters and show how we tune them.

4.3.1 Models in FM

As described in section 2.2, recommendation systems form the basis for question routing. Graphlab has a dedicated library for recommendation system. Factorization machines recommender system is used for our experiments. We use the function -

graphlab.recommender.ranking_factorization_recommender.create *(observation _data, user_id ='user_id', item_id ='item_id', target ='Score', user_data = user's auxiliary features, item_data = question's auxiliary features, num_factors=16, regularization=0.01, linear_regularization =1e-09, side_data_factorization =True, ranking_regularization =0.25, max_iterations = 100, sgd_step_size=0.1, random_seed=0, binary_target=False, solver='auto')*

This Factorization Recommender function learns latent factors for each user and item and uses them to make rating predictions. Observation data is the dataset to be used for training the model. It must contain a column of user ids and a column of item ids. Each row represents an observed interaction between the user and the item. The (user, item) pairs are stored with the model so that they can later be excluded from recommendations if desired. It can optionally contain a target ratings column. All other columns are interpreted by the underlying model as side features for the observations. Side data for user and item can be optionally provided and includes any amount of user or item essential data. All the auxiliary features that we provided were as side data. The expertise of a user is his additional data that is essential for the model to know. Therefore, the various types of expertise are inputted into the model as user and item data. Observation data can have some other attributes (apart from auxiliary features, user and item ids and a target attribute) that may affect the model's predictions, for example, creation date of a question and an answer. There can be any number of additional attributes passed in as observation data. They are treated as latent factors by the model. Pairwise interaction between all features are taken into consideration to create the final model.

4.3.2 Hyper-parameters

In general, a machine learning model is the definition of a mathematical formula with a number of parameters that need to be learnt from the data. That is the crux of machine learning: fitting a model to the data. This is done through a process known as model training. In other words, by training a model with existing data, we are able to fit the model parameters. However, there is another kind of parameters that cannot be directly learned from the regular training process. These parameters express "higher-level" properties of the model such as its complexity or how fast it should learn. They are called hyperparameters.

Hyperparameters[16, 9, 31] are usually fixed before the actual training process begins. Factorization Machines have hyperparameters that can be tuned to suit best for a specific type of data. The most influential parameters are dimensionality of latent factors, regularization and learning rate. They have to be predefined before the model training process.

Dimensionality of latent factors is the number of latent features that the model chooses in matrix factorization. It defines the number of unobservable features to be taken into consideration while predicting an item to a user.

Regularization helps to solve overfitting problem in machine learning. A Simple model will be a very poor generalization of data. At the same time, complex model may not perform well in test data due to overfitting. We needed to choose the right model in between simple and complex model. Regularization helps to choose preferred model complexity, so that model is better at predicting. Regularization is adding a penalty term to the objective function and control the model complexity using that penalty term.

Step size is the learning rate of the model. We needed to find a Rate at which the error was minimum. In the process of training a model, if the step size increases slowly then we might not reach the point of minimum error, whereas if the step size increases drastically then we might miss the minimum and oscillate at a value. The best strategy is to increase the step size exponentially and find a learning rate at which the error in predictions is minimum.

4.3.3 Evaluation Metrics

Since Recommender Systems are the basis for Question Routing systems, their evaluation methods have an overlap as well. A good overview for evaluation Recommender Systems is given in [21]. In this section we discuss some of the evaluation methods mentioned in [21] and will indicate how they are useful for our work.

For this research offline verification is used (others being online and user based verification methods as mentioned in [21]). This approach is chosen because it allows for different system configurations to be tested easily and relatively fast. For offline verification several metrics are available to calculate the performance of a recommendation. These metrics fall into two categories, Set-based and Rank-based. Set-based metrics do not incorporate the order of the recommended items, while Rank-based metrics do.

- *Precision* : It is a Set-based evaluation method. Precision indicates how many of the selected candidates have actually answered the question. Thus it is an indication on how good the suggestions are. Precision is calculated per recommendation. It is calculated as,

$$precision(q) = \frac{A(q) \cap P(q)}{P(q)} \tag{4.5}$$

- *Recall* : It is a Set-based evaluation method. Recall indicates how many of the actual answerers are in the selected candidates set. Thus it is an indication on how well the suggestions represent the full set of answerers. Recall is calculated per recommendation. Recall is calculated as,

$$recall(q) = \frac{A(q) \cap P(q)}{A(q)} \tag{4.6}$$

- *F1* : It is also a set-based evaluation method. F1 score is a combination of both Recall and Precision. F1 weighs both the recall and the precision score, thus provides an indication on how well the suggestions are in general. The suggestions should both have a high precision and a high recall in order for F1 to increase. It is calculated as,

$$F1(q) = 2 * \frac{precision(q).recall(q)}{precision(q) + recall(q)} \tag{4.7}$$

In order to evaluate the overall performance of expertise measurement for question routing using various configurations of recommender system, these values are averaged over all recommendations. Thus given a set of recommendations *R* that have a precision, recall and F1 score, the overall precision, recall and F1 score is defined as follows

$$overallmetric = \frac{1}{|R|} \sum_{\forall q \in R} metric(q) \tag{4.8}$$

- *NDCG* : It is a Rank-based evaluation method that incorporates the rank of the ground truth and the rank of the selected candidates in the calculation. Normalized Discounted Cumulative Gain [22] is a measure that indicates how well the order of the selected candidates is compared to the order of the ground truth. If the order of the suggested candidates is exactly the reverse of that of the order of the ground truth, the score will be much lower than if the order would be exactly the same. If the selected candidates are only a subset of the full ground truth, the NDCG metric can still give the optimal score. For instance if the ground truth is < 1,2,3,4,5 > and the selected candidates are <1,2,3>, the NDCG is optimal. If the selected candidates are <2,1,3> the NDCG is non-optimal. NDCG can thus be seen as a specialized version of precision, although their scores may be very different. The formula for NDCG is based on the Discounted Cumulative Gain (DCG) and the Ideal Discounted Cumulative Gain (IDCG)

$$NDCG_n = \frac{DCG_n}{IDCG_n} \tag{4.9}$$

where, DCG for n is described as,

$$DCG_n(q) = \sum_{i=1}^{n} \frac{2^{rel_i} - 1}{log_2 i + 1} rel_i = (|A_q| - pos_i) + 1 \tag{4.10}$$

For the NDCG measure it holds that if a suggested answerer is not in the ground truth, thus $P_q - A_q \neq 0$ the value for this item is 0. Since NDCG is calculated per question, the NDCG for the Question Routing strategy is calculated as the mean of the NDCG of all questions. Thus for a set of questions Q that are processed the overall NDCG is calculated as

$$NDGC_n^{overall} = \frac{1}{|Q|} \sum_{\forall q \in Q} NDCG(q) \tag{4.11}$$

- *RMSE* : Root Mean Square Error [3] is an error metric for numerical predictions. It is the square root of the mean/average of the square of all of the error. Compared to the similar Mean Absolute Error, RMSE amplifies and severely punishes large errors. It tells us how concentrated the data is around the line of best fit. It is calculated as,

$$RMSE = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - y_i')^2} \tag{4.12}$$

We have included the evaluation metrics that give a fair estimate of performance of different types of expertise for question routing.

4.3.4 Tuning of Hyper-parameters

Using cross validation and grid search, we fine-tune our hyperparameters to suit our data. For experiment, we chose the hyperparameters as :

- Dimensionality of latent factors : [8, 16, 32, 64, 128]
- Regularisation : [0.0001, 0.001, 0.01, 0.1, 1]
- Step Size : [0.0001, 0.001, 0.01, 0.1, 1]

The performance of all three parameters is shown in the figure 4.2.

In the Figure 4.2, to see the effect of dimensionality of latent features in our recommendation, we plot the change in rmse values based on change in latent dimensionality by keeping other hyper parameters as constant. As seen in figure, the error first decreases as we increase the number of dimensions used in factorization, but after a minimum, the error again increases. The error is minimum at 16 dimensions. This indicates that our data is complex, but not complex enough to require large number of features for predicting a question to a user. Similarly, to see the effect of regularization in our recommendation, we plot the change in rmse values based on change in regularization by keeping other hyper parameters as constant. As seen in figure, the error first decreases as we increase the regularization value used in factorization, but after a minimum, the error again increases. The error is minimum at 0.01. This indicates that our data does not require neither a highly complex model, nor a very simple model. A semi complex model will be sufficient. Lastly, in figure to see the effect of step-size in our recommendation, we plot the change in rmse values based on change in learning rate by keeping other hyper parameters as constant. As seen in figure, the error first decreases as we increase the step size used in factorization, but after a minimum, the error again increases. The error is minimum at 0.1. This indicates that our data requires high learning rate. On combining the above analysis, we set our hyperparameters to [16, 0.01, 0.1].

4.4 Configurations

To analyse the performance of our expertise measurement for question routing, an experiment with different configurations of recommender system was conducted. Once the hyperparameters were tuned, the next step was to create a baseline model using factorization recommender and then add auxiliary features later on.

Our baseline included:

1. Question routing with no expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data.

Further configurations include some traditional ways of measuring expertise as auxiliary features-

2. Question routing with existing measures of expertise - Reputation : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition to the baseline, *reputation* of a user was added as a user side data to be included in the pairwise interaction of FM.

3. Question routing with existing measures of expertise - MEC : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition to the baseline, *MEC* of a user was added as a user side data to be included in the pairwise interaction of FM.

4. Question routing with existing measures of expertise - Reputation + MEC : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition to the baseline, *reputation* and *MEC* of a user was added as a user side data to be included in the pairwise interaction of FM.

5. Question routing with existing measures of expertise - Reputation + MEC + Tags : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition to the baseline, *reputation*, *MEC* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.

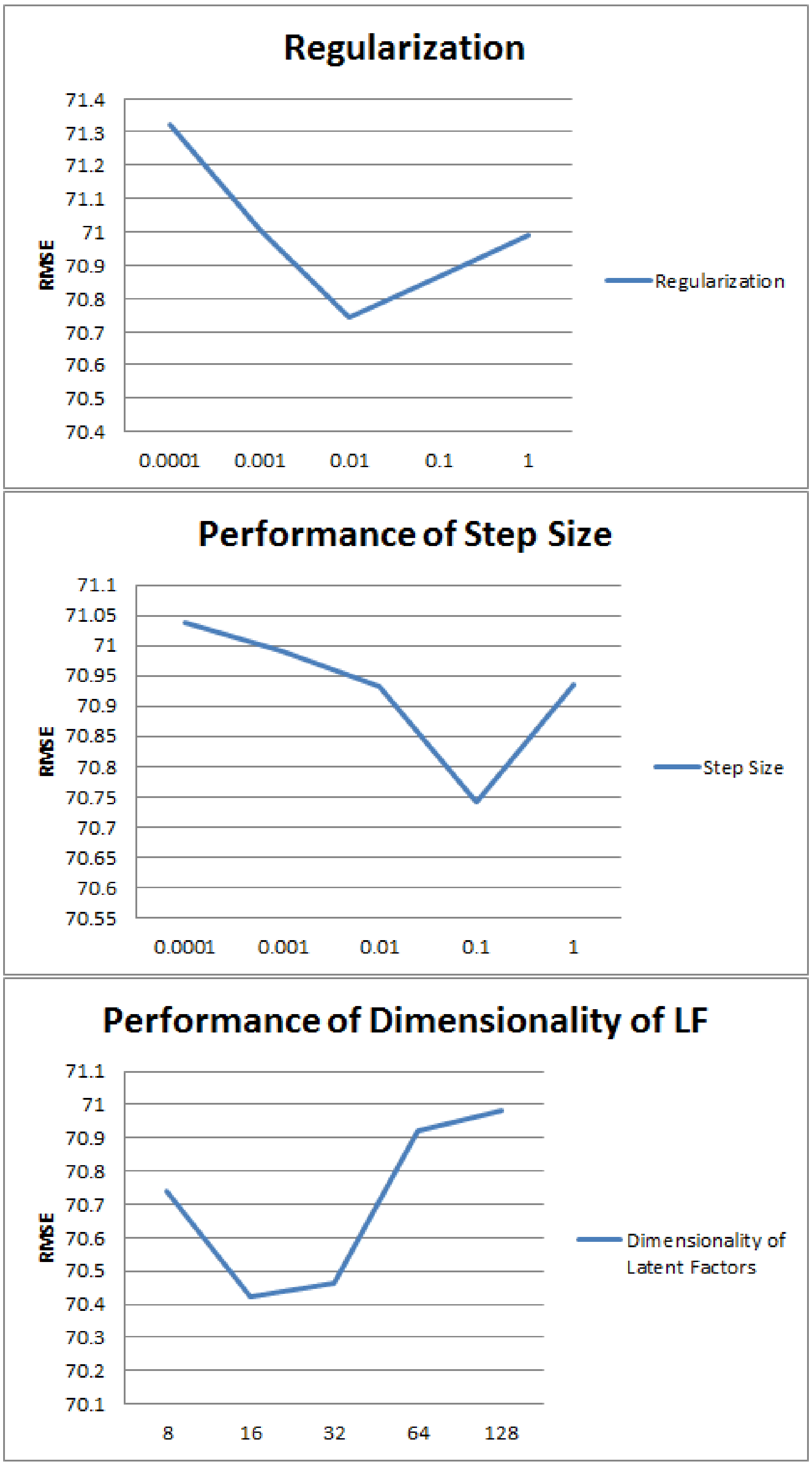


Figure 4.2: Performance of Dimensionality of Latent features, regularization and step-size. *We can see with increase in value, the RMSE values first decreases, comes to a minimum and then increases.*

Configuration with which we test our expertise measurement techniques as auxiliary features include:

6. Question routing with social expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition, *SoE* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.
7. Question routing with ubiquitous expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition, *UE* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.
8. Question routing with specialist expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition, *SpE* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.
9. Question routing with specialist expertise + ubiquitous expertise + social expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition, *SoE*, *SpE* and *UE* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.

In order to see the performance of our techniques in conjugation with traditional methods, we added a final configuration with all auxiliary features.

10. Question routing with all types of expertise : This configuration included *ownerid* of the answer as a user, *parentid* of the answer as the item, *score* as the target value and *timestamp* as additional data. In addition, *reputation*, *MEC*, *SoE*, *SpE* and *UE* of a user as a user side data and *tags* of a question as item side data was added to be included in the pairwise interaction of FM.

The Precision, recall and rmse values were recorded for all configurations. The results of our experiments are shown and discussed in the next chapter.

In order to conduct experiments to compare performance of above configurations, we first conduct experiments to see the best possible way of dividing our data into training and test datasets. We divide our data temporally (experiment included in next chapter).Then, we designed our experiments with different data sets.

1. 4 sets of data -
   - Answers within a time period of 1 month
   - Answers within a time period of 6 month

- Answers within a time period of 1 year
- Answers within a time period of 2 year

2. Division of Training and Test Data

   - 50% test and 50 % training
   - 60% test and 40 % training
   - 70% test and 30 % training
   - 80% test and 20 % training

This was done to get a fair estimate of performance of expertise measurement on varying datasets.

4.5 Chapter Conclusion

In this chapter we address our two research questions (RQ1 and RQ3). We answer our first research question by defining different types of expertise and what they are. We map user behaviour with different types of expertise and present the measurement technique for each type of expertise. We then move further to answer our third research question. We introduce the model used in FM, the concept of hyperparameters within FM and tune them. We then discuss various evaluation metrics like precision, recall, rmse, ndcg that will be used in evaluating the performance of our experiments.

In the last section we present all configurations used in experiments, results of which are showed in the next chapter.

Chapter 5

Results and Discussion

In order to demonstrate the effect of our proposed expertise measures, we first empirically analyse the dataset that we constructed and then apply it to question routing. In order to do this, we conducted a series of experiments. These experiments eventually provide an empirical support to our answers for RQ1 and RQ3. In this chapter we first show the overview of data and how we divide our training and test data. In the next section we present the readout for mapping user behavior. Then we present the readout for applying expertise measure to question routing. We discuss the results and threats to validity in the end.

5.1 Overview of Data

5.1.1 Division of Training and Test Data

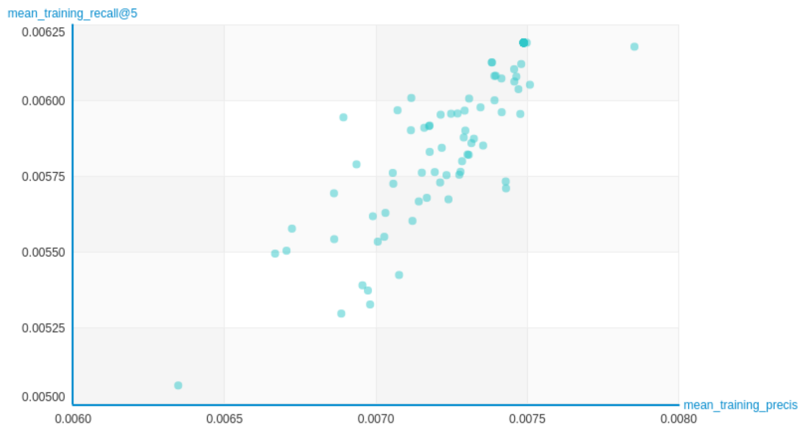
In our experiment, from stackoverflow we took questions as items and owners of answers as users (uniquely represented by each answer). The actual score of an answer was used as a rating.

Figure 5.1 shows two comparable graphs. Figure 5.1a represents models plotted when data is divided into test and training data as per 5-fold cross validation. Figure 5.1b represents models plotted when data is divided into test and training data as per holdout method, i.e. 50% data as test and 50% data as training set. We see a difference in both graphs with respect to clustering of models. Models (precision, recall values) are more scattered when divided by fold method. This is due to the fact that our data set is temporal in nature. When the data is divided temporally into 50-50 training-test sets, the recommendations perform better. Time was an important factor for our analysis. Every question and answer had an associated timestamp.

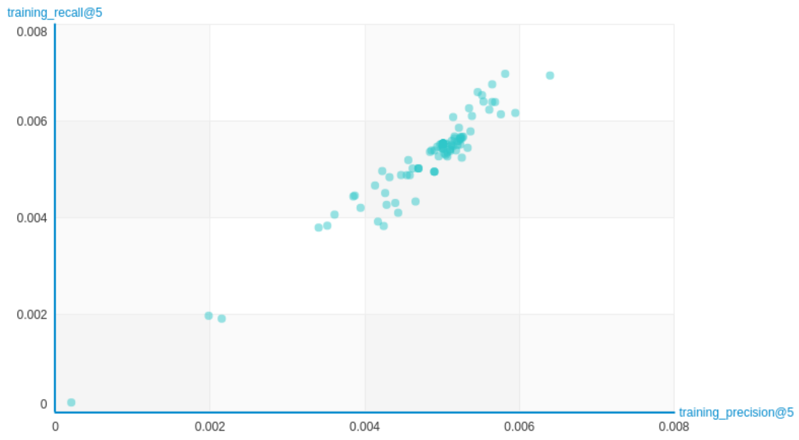
	Precision	**Recall**	**RMSE**
Recommendation with 5-fold cross validation	0.0042754	0.0057554	81.682
Recommendation with holdout method (50-50)	0.00639	0.00694	71.083

Table 5.1: Evaluation of division of training and test data

Further, the Table 5.1 justifies this point. It shows that precision and recall of a recommendation model with training and test data divided using holdout method are



(a) Models plotted after 5-fold cross validation



(b) Models plotted after holdout method cross validation

Figure 5.1: Comparison between models for different data division for evaluation

greater than that of recommendation model created using 5 fold cross validations. We get this result because a user that joined stackoverflow, after a question was posted, answered and then finally closed, was not eligible to answer that question. Therefore k-fold cross validation was not a good option. However, we experimented creating a recommendation model using both 5-fold cross validation and 50 -50 division of data into training and test sets.

Once we knew that data had to be divided according to *creationdate* of a question and answer, we experimented to select our optimum sample size. We divided the dataset into small sets consisting of all questions and answers posted within 2 years, 1 year, 6 months and 1 month. The distribution of number of answers for a question is shown in figure 5.2 for all datasets. There are a total of 4,770,694 questions and 3,849,827 answers in 2 year dataset; 2,584,051 questions and 2,706,288 answers in 1 year dataset; 1,332,057 questions and 1,610,429 answers in 6 month dataset; and, 219,817 questions and 248,399 answers in 1 month dataset. The mean of number of answers for each question varies from 0.9497 to 1.197. The standard deviation ranges

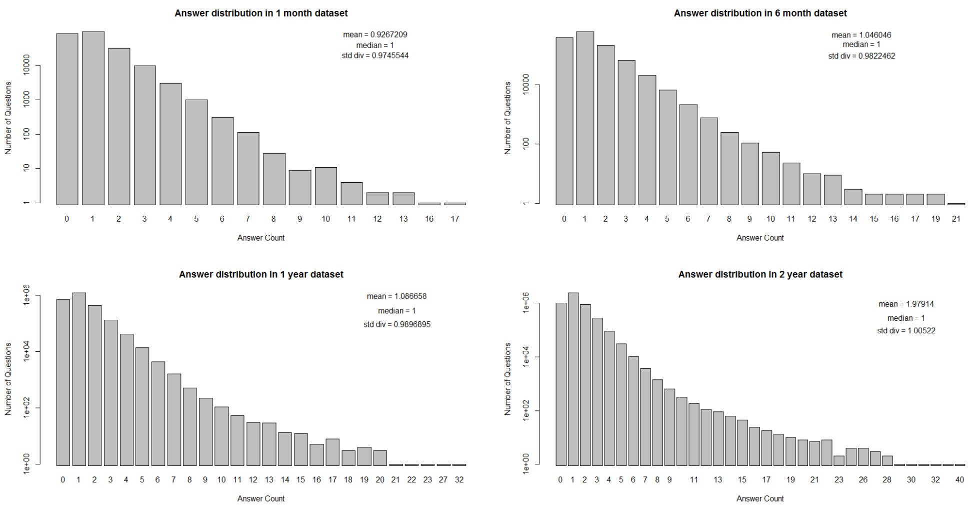


Figure 5.2: Distribution of Number of Answers per Question in different datasets. Y axis shows a logarithmic scale in all plots.

	1 month	6 months	1 year	2 year	Full dataset
minimum	-16	-29	-29	-29	-65
maximum	327	638	1866	1866	21403
mean	0.7076	0.8137	0.9069	1.0707	2.6492
median	0	0	0	0	1
standard deviation	1.7158	2.1781	2.8234	4.1114	19.4630
variance	2.9441	4.7443	7.9715	16.9036	378.8092

Table 5.2: Quantitative measurements of *Score* earned by all answers within the stipulated dataset sizes.

from 0.9745 to 1.005. These values suggest that the majority of the questions receive only 1 answer. Also, there are a large number of unanswered questions as we can see in the graphs 5.2.

Table 5.2 shows the statistics of *score* values of answers in all datasets. *Score* values are really important for this study as they are the measure of rating an answer gets from other users (through up-votes and down-votes), hence a reflection of relevance of an answer for the question. From the table 5.2 we can see that the mean value is ranged between 0.7 to 2.6. This means that the majority of the answers receive an equivalent of single up-vote.

Moving forward, next in order to measure and study the proposed expertise described in sections 4.2.3, 4.2.4 and 4.2.5, we conduct our experiments on the basis of topic. There are a total of 5299 topics or tags in our data. Table 5.3 shows a list of ten most popular tags in our database. *javascript* being the most popular tags and with maximum number of associated questions, we chose javascript as our topic of study. In all our further experiments we measure expertise of a user for the topic *'javascript'*. Based on the final results we can apply expertise measurement in a generic manner for all topics.

Tag Name	Tag Count
javascript	1466567
java	1309191
c#	1134064
php	118654
android	1026116
jquery	867220
python	813903
html	687658
c++	531938
ios	527365

Table 5.3: Top 10 tags and their count in our database

Total number of questions which have a tag 'javascript'	1466567
Total number of answers belonging to questions that have a tag 'javascript'	2401440

Table 5.4: Statistics of tag *javascript*

We choose the 1 year dataset to conduct our next set of experiments for calculating expertise of a user on the topic '*javascript*'. The 1 year dataset has 301,678 questions associated to '*javascript*', and 322,313 answers associated with those questions. This gives us a decent amount of data to create recommendation models on.

5.2 Readout for Mapping user behaviour

Expertise can only be measured when we know what actions of a user on web platforms reflect his mastery in the field. These actions are called as actionable knowledge in [49] and we continue to call them the same. In order to map a user behaviour to a type of expertise as described in sections 4.2.3, 4.2.4 and 4.2.5, we first look at some user actions showcased within our data.

Distribution of users having code snippets in answers : Figure 5.3 shows the distribution of code snippets (log-log plot) included by users in their answers. It has an exponentially decreasing curve on a logarithmic scale. This means that answers with higher number of code snippets are very less. This is also backed by the high value of variance (249.21). The mean being 3.559577 and standard deviation being 15.78657 prove that the majority of the users tend to include three pieces of code in their answers. The high value of maximum (1202) show a that there are very few answers that have large number of code pieces incorporated in them.

Distribution of Comments and Comments with code snippets : Figure 5.4 shows the log-log distribution of comments in an answer and comments with code snippets in an answer. Figure 5.4a shows the distribution of comments. It has an exponentially decreasing curve. This means that very few answers have a high number of comments. This is absolutely logical. However the average number of comments in an answer is nearly 200. This was relatively higher than what we had expected. The high mean value is probably recorded due to limited sample size, and is expected to reduce

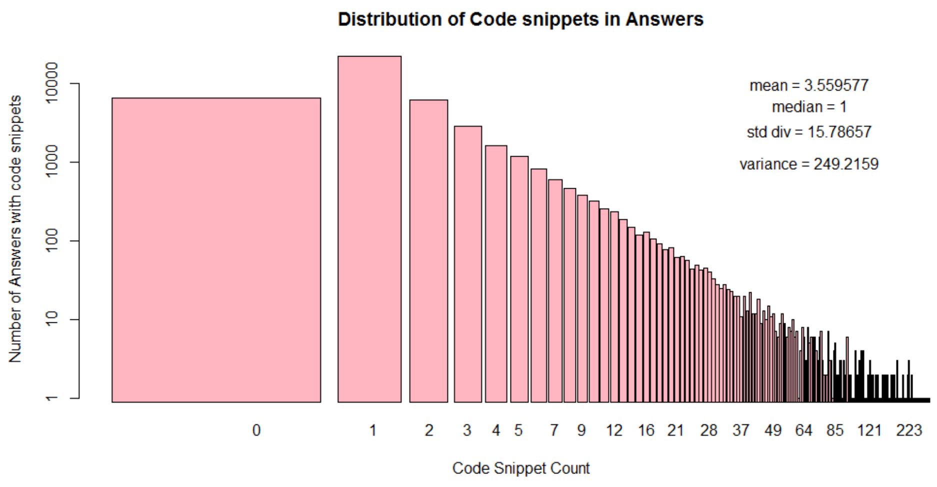


Figure 5.3: Distribution of code snippets included by users in their answers. *Both the x and y axis show a logarithmic scale.*

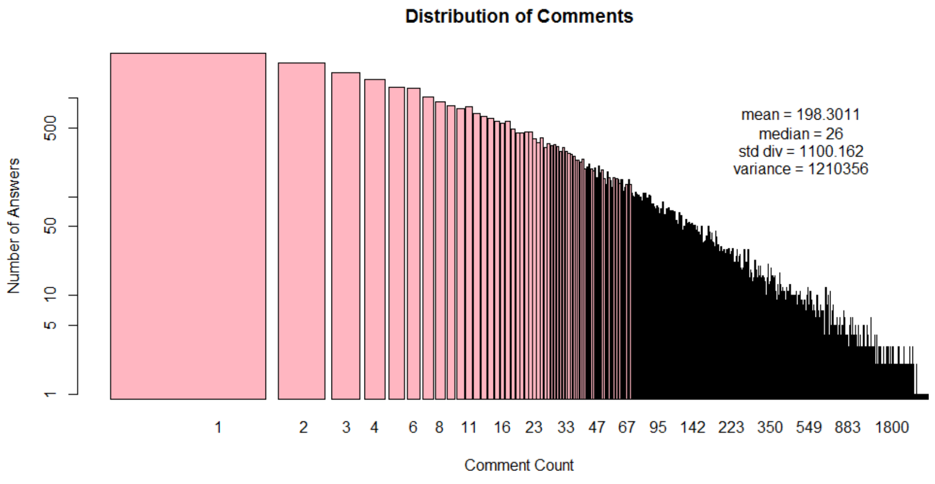
as the sample size increases. Figure 5.4b shows distribution of code snippets in comments. The median value of 0, is an expected result, as majority of the comments do not include code. Comments are used to provide an explanation and clarifications for the underlying answers. Therefore it is logical that fewer comments have code snippets. If we did a supporting analysis of length of codes incorporated in a comments, this would have been much lesser than the average length of code snippet in an answer.

Number of answers having a Github link : Figure 5.5 shows a graph of distribution of explicit Github Links in a Stack-Overflow Answer. The median of 0 shows that majority of the users do not mention a github link in their answers. A mean of 0.145 also supports this statement. The thickness of the bars suggest the density of Github link count. The user behaviour of providing an explicit Github link in their answers is the most important action associated with a specialist expertise. When we calculate Specialist Expertise, the correlation between expertise value of a user and his action of providing explicit Github link give value of 0.572 . This value is of 95% confidence and with a p-value of 0.002, which shows the significance of the result.

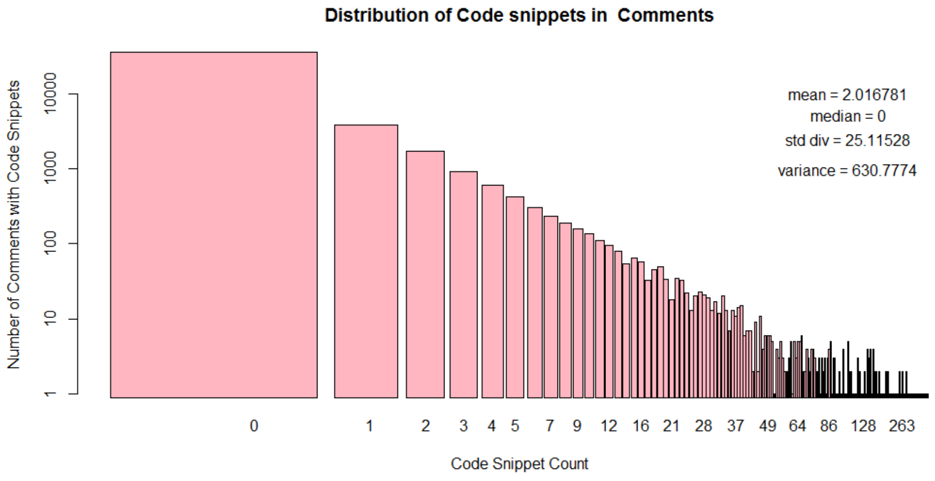
Apart from the above mentioned user behaviors, we also quantify other actionable knowledge of a user like - Answers with links to own Github account, Tweets with Github links, Tweets on topic *javascript*, Re-Tweets with Github links, etc. With all user behaviors being mapped and inputted in the equations mentioned in sections 4.2.3, 4.2.4 and 4.2.5, we calculate the Ubiquitous, Specialist and Social Expertise of a user. Further subsections of this section gives more insights into the measured expertise of a user.

5.2.1 Correlation between different types of Expertise

We calculated the correlation between three types of expertise.



(a) Distribution of Comments



(b) Distribution of Comments with Code

Figure 5.4: Log-Log Distribution of Comments and Comments with Code in Answers

Correlation between Ubiquitous Expertise and Specialist Expertise The Pearson product-moment correlation between Ubiquitous expertise and specialist expertise results in:

p-value < 2.2e-16

95% confidence interval : 0.3804509 and 0.3972889

The Sample estimate correlation : 0.3888923

This positive correlation of 0.38 and a very small p-value shows that linear relationship exists between the two types of expertise, and this result is a significant result. However, the linear relationship is not a perfect linear relation, but a moderately positive linear relation.

Correlation between Ubiquitous Expertise and Social Expertise The Pearson

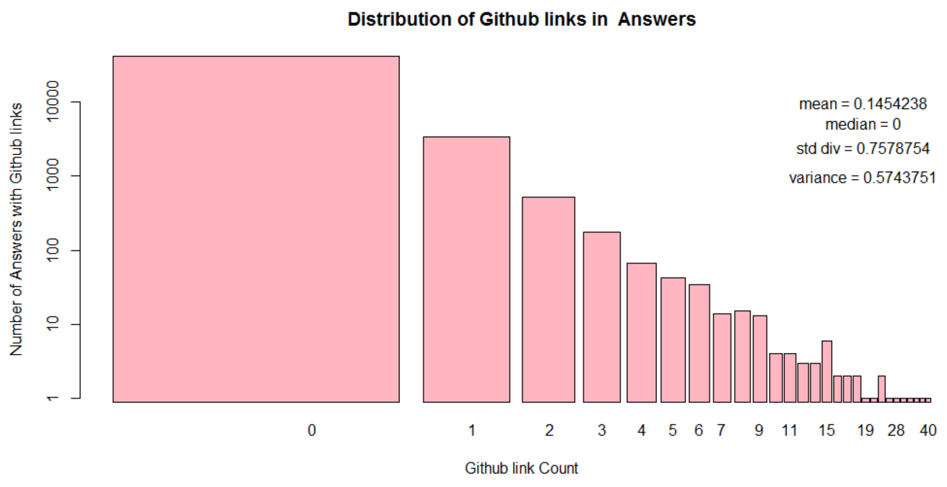


Figure 5.5: Log-log Distribution of Github Links in a Stack-Overflow Answer.

product-moment correlation between Ubiquitous expertise and Social expertise results in:

p-value < 2.2e-16

90% confidence interval : 0.1478922 and 0.1512972

The Sample estimate correlation : 0.1487933

This positive correlation of 0.14 and a very small p-value shows that linear relationship exists between the two types of expertise, and this result is a significant result. However, the linear relationship is quite small and close to no relation.

Correlation between Social and Specialist Expertise The Pearson product-moment correlation between Specialist expertise and Social expertise results in:

p-value < 2.2e-16

93% confidence interval : 0.5482392 and 0.5497162

The Sample estimate correlation : 0.5487913

This positive correlation of 0.54 and a very small p-value shows that linear relationship exists between the two types of expertise, and this result is a significant result. The linear relationship is moderately high and makes sense because we calculate social expertise based on a user's friendship with other specialist experts.

The correlation between Social and Specialist expertise is the highest at 0.54, followed by Ubiquitous and Specialist expertise at 0.33 and Ubiquitous and Social expertise being least at 0.14. All the three correlations values between the three types of expertise are positive. This shows that some form of dependence exists between them. The relative high value of correlation between SpE and SoE is due to the fact that social expertise is based on the existence of social bonds between a user and a specialist user. Therefore their ought to be a high correlation between the two. UE and SpE have a moderately high correlation. This is in accordance to our working hypothesis, that if a user is a specialist on a topic, he will also showcase signs of high level of ubiquitous

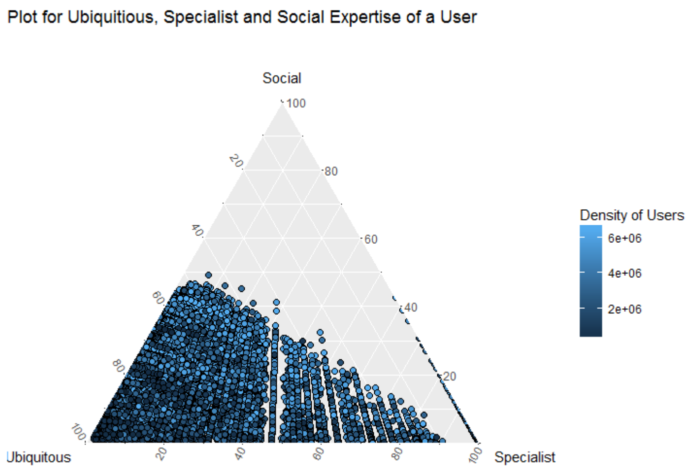


Figure 5.6: Plot for Ubiquitous, Specialist and Social Expertise. *The three scales represent percentage of values*

expertise; but vice verse is not true, an ubiquitous expert will not automatically be a specialist expert. The low positive correlation between Ubiquitous and Social expertise can be justified by stating the fact, that, as per definition of Social expertise, it depends on the existence or non-existence of Specialist experts as friends, they have no influence by the existence or non-existence of Ubiquitous experts as friends. Therefore, if a user is an Ubiquitous expert he may or may not be a Social expert. These two types of expertise are not inter-dependent.

5.2.2 Measures of different types of Expertise

The Figure 5.6 shows a ternary plot for Ubiquitous, Specialist and Social expertise of a user. The three sides represent each type of expertise with percentage values increasing from 0 to 100. Each point represents a human as point with the three types of expertise normalised to add to 1, and the color of the points shows its density. The lighter the color, more number of users with that value of expertise.

From the figure, we can see that the concentration of all users is in left corner. This shows that the maximum number of users have more ubiquitous expertise than specialist or social expertise. Further, the concentration of all users in lower half of the triangle, suggests that, their is no user with high social expertise values. This is due to the fact that our measurement of social expertise includes friendship with a specialist expertise, and this is really hard to capture. Another observation from the graph is that, their is no user with a perfect 100% ubiquitous score. Also, few users who have high specialist expertise values showcase social expertise. The high concentration of users with higher Ubiquitous expertise and low specialist expertise (hence lower left corner) is supported by the fact that in our data, most users had code snippets in their answers (indicator of ubiquitous expertise) but very few github links (indicator for specialist expertise). Further, the absence of any user point in the upper half of the triangle towards social expertise is backed by the fact that *javascript* topic is not an actively tweeting user-base, it is a reading user-base (more number of accounts as compared to number of tweets and re-tweets). Hence we were not able to capture any user with high value of Social Expertise.

All the above mentioned observations from the figure 5.6 justify our arguments related to all three types of expertise measurements. In the next section, we apply this measured expertise of users to question routing.

5.3 Readout for Applying expertise to QR

Once we had our expertise measurement technique and calculated expertise for our sample data users, next step was to apply this expertise characterization on question routing. As described in the previous chapters, recommendation systems are an effective way to evaluate the performance of question routing service. We use Factorization Machine due to its peer-to-peer interaction latency learning capabilities. Section 4.4 introduces all configurations configured to evaluate the performance or our recommendation system.

The 1 year dataset for topic *javascript*, divided into 50% test and 50% training data was used in all configurations. The Precision, recall and RMSE values were recorded for each of them. The table 5.5 shows a comparison between the ten configurations.

Configurations	**Precision**	**Recall**	**F1**	**RMSE**
Baseline	0.00020801	0.0004645	0.0002501	3.058
Baseline + Reputation	0.0004087	0.0004469	0.0004269	3.061
Baseline + MEC	0.0004121	0.000538	0.0004667	3.058
Baseline + Reputation + MEC	0.00051813	0.000541	0.0005293	3.155
Baseline + Reputation + MEC + tags	0.00050798	0.000691	0.0005855	2.954
Baseline + SoE	0.00038109	0.0005645	0.000455	3.063
Baseline + UE	0.00076219	0.0005734	0.0006544	2.066
Baseline + SpE	0.0006815	0.0005782	0.0006256	2.0614
Baseline + SoE + UE + SpE	0.00065813	0.0005968	0.00062596	2.017
Baseline + all	0.00078109	0.00068245	0.0007284	1.7491

Table 5.5: Comparison between different configurations

From the above table, we can see that the performance of the model increases (decrease in RMSE values and increase in precision recall value) when we add MEC and tags with reputation and score. MEC or mean expertise contribution indeed proves to be a good estimate of a user's expertise. Previous work suggested the use of MEC with Reputation score as a method for improving the performance of question routing. However in our work, when we introduce the Social, Ubiquitous and Specialist expertise, we can see a further improvement in the performance.

The system does not perform well (RMSE - 3.063) when we run it with just the baseline and social expertise. This is according to our expectations, because social expertise is an add along measure and is not self sufficient. It represents a users' social behavior with other expert users and does not guarantee the expertise of a user himself.

The performance of the system with Ubiquitous expertise and baseline is a major improvement from previous configurations. Its precision value is amongst the best. We

Configuration	Accuracy
Baseline + Reputation + MEC	76.3%
Baseline + SoE + UE + SpE	77.64%
Baseline + all	78.03%

Table 5.6: Accuracy of Models

can clearly see that Ubiquitous expertise is a better performing expertise measure than MEC which was an improvement in itself. The configuration with Specialist expertise also performs well with an RMSE value of 2.06. However when we combine all three types of expertise proposed in this work, the system performed well but we do not find any drastic improvement. We did not expect that. This result showed that our three types of expertise, together do complement each other but at low levels. Had their been major improvements in the precision recall values, then we could have said that Ubiquitous, Specialist and Social expertise highly compliment each other and should be used together.

Our last configuration included all forms of expertise, our proposed measures along with previously proven methods. We can clearly see that this configuration performs best. It has the lowest RMSE value and highest Precision, Recall and F1 values. This was an expected result. The RMSE value of 1.7491 is the closest to the actual mean value of *score* (0.906) for this dataset.

We did not calculate NDCG values, as the focus of this study was to show the effectiveness of proposed cross-platform expertise measures, and that can be easily evaluated using Precision, Recall and RMSE values. NDCG is usefull when we evaluate the ranking of expert users recommended for a question using different configurations. But this study is out of scope for our work. However, we calculated the Accuracy of three of our configurations. Table 5.6 shows the Accuracy. The Accuracy for all three configuration lie within the range of 76-78%. However, models with cross-platform expertise have slightly more accurate results than previously proposed expertise methods. Although the improvement is only of 1.7%, yet this is a major improvement when it comes to applicability of the model in real world scenario.

5.4 Discussion

By incorporating our proposed expertise measures along with reputation and mean expertise contributions of a user, we add more certainty that the users' recommended by this model would be ranked in accordance to their actual knowledge.

We observed from our results that although Ubiquitous and Specialist expertise can be self sufficient to improve the performance of community question answering service such as Stack-Overflow, Social expertise is not an adequate enough measure to improve the performance on its own. The cross-platform expertise measurement such as Ubiquitous and Specialist expertise and MEC which is derived from the question answering platform itself, together really improve the performance of the question routing platform. This is evident from the Accuracy Value of our prediction being at 78%.

A human user exhibits certain amount of all types of expertise. The value of one type of expertise may be bigger for one topic but almost negligible for another. Within question routing, a question can belong to more than one topic simultaneously, thus it is necessary that the system should decide what type of expertise is more important and suggest recommendations accordingly. This is a topic to be considered in future work. However, here in our work, for us it is important to make sure that the recommended ubiquitous experts are actually ubiquitous experts and specialists are actually specialist experts with the right kind of specified actionable knowledge. A model which adopts our final configurations consisting of all types of expertise may be favourable in situations where a generic form of expertise needs to be mapped, but un-favourable in situations where, for example, only highly specialist experts are required. Having said that, as compared to generic expertise derived from single platform, a user expertise derived from within a cross-platform framework is a better model to adopt.

Expertise of a user, mapped according to his behavior on the social web on multiple platforms is indeed a better characterization of his actual knowledge levels as compared to that measured from the question answering platforms itself. The problem of favouring activeness of a user is also reduced if the systems use our proposed measures. The proposed measurement techniques can be further fine tuned to reflect even sharper view of a users' knowledge by incorporating even deeper analysis of the platforms under consideration.

5.5 Threats to Validity

In this section, we discuss the limitations in our cross platform expertise recommendation for question routing. We also discuss the threats to validity for the approaches we designed for high accuracy user data collection and matching and estimating reliability of measuring expertise of a user in cross platform setting.

5.5.1 Cross Platform expertise characterization for question routing

We mainly focused on expertise characterization of user and provided a method for expertise characterization with cross-platform data. We did not focus on the process after the calculation of expertise till expert users are recommended to a question. Though we provided a proof of concept of our question routing by implementing a recommendation model, it was not evaluated with live user requests.

5.5.2 User Matching

We designed a pipeline to collect high accuracy matching users. We collected the possible user profiles from Stackoverflow, Github and Twitter. We performed three step matching including direct, accurate and fuzzy matching. Attribute matching was limited due to the unavailability of emailhash of new users on both stackoverflow and github. Gravatarid hash was also limited for new users by the fact that we had to parse it based on profile pictures, and not many users had a profile picture.

Fuzzy Matching was limited by the restrictions of Twitter's REST API. We could only process 100 users per request for *lookup* and 1 user per request for *search*. Due to this reason, we had to limit our Stackoverflow and Github datasets. Further, the user matching pipeline included mutations in the *loginnames*. If we had applied more complex and bigger mutations, maybe we could have matched more number of users and thus extending our user base.

5.5.3 Expertise Measurement

Expertise measurement had limits to its validity due to the fact that there were limited actionable knowledge actions that we could derive from our data. If we had taken into consideration comments or actual contributions in a github repository, we could have increased the accuracy of our expertise calculation. The formulas derived for Ubiquitous, Specialist and Social expertise suffered from the lack of experimentation. We did not experiment on the method for measuring each type of expertise. We could have maybe found a better performing measure.

The experiment for evaluating the performance of recommendation system was conducted only for one topic. We did not experiment for other topics. This limits the validity of the proposed method, however we are certain that our proposed methods will perform on other topics as well but is limited to the scope of software development. We cannot generalize our work to other domains of knowledge. However, software development being a representative form where expertise can be of multiple types, and where the Web plays an important role in expertise development. This makes the experimental evaluation described in this work potentially usefull for supporting similar research targeting domain with knowledge having similar characteristics.

Chapter 6

Conclusion and Future Work

This chapter gives an overview of the project's contributions. Then, summarizes the work done and all findings. In the end we present the future scope of this work.

6.1 Conclusion

In this thesis work, we research about efficiently improving expertise characterization for question routing using cross-platform data. To solve this main research question we started with understanding the architecture of community question answering services, the challenges that characterize the routing of a question to an expert answerer, training and execution of recommender system as a way to implement question routing, and the opportunities and limitations in using human cross platform user data to support the expertise measurement. We studied details of these areas thoroughly in our literature survey. An extensive literature study provided us insights on how the question routing systems with multi web collaborative platform data also provided us guidelines to design such a question routing system.

For characterizing expertise with data from multiple web collaborative platforms, we first identified the types of expertise existing within the domain, and designed the method of measuring them for three types of expertise - ubiquitous, specialist and social. We presented how incorporating cross-platform expertise could improve the performance of question routing recommender system. To prove the feasibility of our envisioned system, we conducted a series of experiments using stackoverflow, github and twitter data for software development domain.

Before we conducted the experiments to find the best configuration for question routing recommender system, we focused on creating user model within cross-platform framework. For this, we first collected maximum feasible user data from multiple web collaborative platforms - stackoverflow, github and twitter. We designed an approach to match user profiles across network with high accuracy. We showed that the approach achieved high accuracy in matching user profiles belonging to the same user on all three platforms.

After collecting high accuracy user data, we tuned the hyperparameters of factorization machine based on our data. The model was then tested for different data set sizes configurations and predictions were calculated. The trend of performance for different training and test data sizes were observed and analyzed. The validity of approach and results were discussed and they were relevant and beneficial. In the end we showed how a recommendation model for routing a question to a user with expertise on the topic, can be improved by incorporating auxiliary features that depict some form of expertise. The auxiliary features adhere to the cross-platform framework.

6.2 Contributions

By automating the process of measuring expertise of a person for a question, the question routing services is improved. If an expert user will be able to find relevant questions to answer, and the questions will get high quality answers, the QA service will reduce the cost of investment of time required by an expert to find a suitable question, and hence improve the overall productivity of question answer system.

The contributions provided by us in this piece or work include,

- Literature survey of expertise characterization for question answer routing.
- Data Collection Pipeline from multiple web platforms to build user profiles. This was our answer to **RQ2**.
- Framework of expertise characterization using data from multiple web collaborative platforms. A part of **RQ1** was answered by this. We determined the types of expertise existing within the domain.
- Calculating cross platform expertise using data from web collaborative platforms. This was an answer to the remaining part of **RQ1**. We did an extensive literature review and added to it our own understanding of the subject matter, to build a novel method of measuring expertise.
- Evaluation of performance of recommender systems for question routing with different configurations. This was an answer to our **RQ3**.

6.3 Future work

Community Question Answering websites produce other types of meta-data for the posted question and answers such as *favourite count*, *last edit data*. Moreover, user information often contains meta-data information such as *badges*, *reputation*. Using this additional information may help improve the performance of expert recommendation system in future. Deepening our analysis of user interaction network properties such as formation and evolution of communities, and understanding the topic being discussed across three platforms may further improve the performance.

As a future work, we can broaden the scope of expertise characterization by taking into account other social platforms and addressing the challenges of cross-platform user expertise based on complex network models.

Bibliography

[1] L. von Ahn and L. Dabbish. Designing games with a purpose. *Commum. ACM*, 51(8):58–67, 2008.

[2] L. A. Amamic, J. Zhang, E. Bakshy, and M.S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In *17th International Conference on World Wide Web, WWW'08*, pages 665–674. ACM, 2008.

[3] Andale. Rmse: Root mean square error.

[4] Huttenlocher D. Kleinberg J. Leskovec J.: Kleinberg J. Anderson, A. and Leskovec J. Steering user behavior with badges. In *22nd International Conference on World Wide Web, WWW '13*.

[5] Anietie Andy, Satoshi Sekine, Mugizi Rwebangira, and Mark Dredze. Name variation in community question answering systems. pages 108–117. Proceedings of the 2nd Workshop on Noisy User-generated Text, 2016.

[6] A. Badashian, V. Esteki, A. Gholipour, A. Hindle, and E. Stroulia. Involvement, contribution and influence in github and stack overflow. In *CASCON '14 Proceedings of 24th Annual International Conference on Computer Science and Software Engineering*, pages 19–33. ACM, 2014.

[7] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '06*, pages 43–50. ACM, 2006.

[8] K. Balog, L. Azzopardi, and M. De Rijke. Formal models for expert finding in enterprise corpora. In *29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 43–50, 2006.

[9] Remi Bardenet, Matyas Brendel, Balazs Kegl, and Michele Sebag. Collaborative hyperparameter tuning. In *30th International Conference on Machine Learning, Atlanta, Georgia, USA, 2013*, volume 28. JMLR: WandCP, 2013.

[10] Y. Benkler. Coase's penguin, or, linux and the nature of the firm. *Yale Law Journal*, 112(3):367–445, 2002.

[11] A. Berger, R. Caruana, D. Cohn, D. Freitag, and V. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '00*, pages 192–199. ACM, 2000.

[12] Y. Cao, H. Duan, C. Yew Lin, and H. Wuen Hon. Recommending questions using the mdl-based tree cut model. In *17th International Conference on World Wide Web, WWW '08*, pages 81–90. ACM, 2008.

[13] M. T. Chi, R. Glaser, and Farr M. J. The nature of expertise. In *Psychology Press*, 1998.

[14] H. Collins and R. Evans. *Rethinking Expertise*. University of Chicago Press, 2007.

[15] B. Dom, I. Eiron, A. Cozzi, and Y. Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In *8th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 42–48. ACM, 2003.

[16] Kaibo Duan, S.Sathiya Keerthi, and Aun Neo Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:41–59, 2003.

[17] Dag Erling. Allow non-alphabetic characters in user names.

[18] Y. Fu, R. Xiang, Y. Liu, M. Zhang, and S. Ma. A cdd-based formal model for expert finding. In *Sixteenth ACM Conference on Information and Knowledge Management, CIKM '07*, pages 881–884. ACM, 2007.

[19] J. Guo, S. Xu, S. Bao, and Y. Yu. Tapping on the potential of q and a community by recommending answer providers. In *17th ACM Conference on Information and Knowledge Management, CIKM '08*, pages 921–930. ACM, 2008.

[20] Benjamin V. Hanrahan, Gregorio Convertino, and Les Nelson. Modeling problem difficulty and expertise in stackoverflow. In *ACM 2012 Conference on Computer Supported Cooperative Work Companion, CSCW '12, New York, NY, USA*, pages 91–94. ACM, 2012.

[21] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. E. Evaluating collaborative filtering recommender systems. In *ACM Trans. Information Systems, January 2004*, 2004.

[22] Sijmen Peter Hoogendijk. Ask the right expert.

[23] Johnny Huang. Image duplication detection.

[24] Paul Jaccard. The distribution of the flora in the alpine zone.1. *New Phytologist*, 2(11):37–50, 1912.

[25] S. Jain and D. Parkes. Designing incentive for online question and answers forums. In *10th ACM Conference on Electronic Commerce, ACM '09*. ACM, 2009.

[26] J. Jeon, W. B. Croft, and J. H. Lee. Question answering from frequently-asked question files: Experiences with the faq finder system. In *Technical report, AI Magazine*, 1996.

[27] J. Jeon, W. B. Croft, and J. H. Lee. Finding semantically similar questions based on their answers. In *28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 617–618. ACM, 2005.

[28] J. Jeon, W. B. Croft, J. H. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *29th Annual International ACM, SIGIR '06*, pages 228–235. ACM, 2006.

[29] Zongcheng Ji and Bin Wang. Learning to rank for question routing in community question answering. In *22Nd ACM International Conference on Conference on Information and Knowledge Management, CIKM'13*, pages 2363–2368. ACM, 2013.

[30] Wei-Chen Kao, Duen-Ren Liu, and Shiu-Wen Wang. Expert finding in question-answering websites: A novel hybrid approach. In *2010 ACM Symposium on Applied Computing, SAC '10, New York, NY, USA*, pages 867–871. ACM, 2010.

[31] S.S. Keerthi. Efficient tuning of svm hyperparameters using radius/margin bound and iterative algorithms. *IEEE Transactions on Neural Networks*, 13(5):41–59, 2002.

[32] Yang L., Bao S., Lin q., Wu X., Han D., Su Z., and Yu Y. Analyzing and predicting not-answered questions in community-based question answering services. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 1273–1278. AAAI, 2011.

[33] Liang-Cheng Lai and Hung-Yu Kao. Question routing by modeling user expertise and activity in cqa services. In *The 26th Annual Conference of Japanese Society for Artificial Intelligence, 2012*, volume 3M1-IOS-3a-2. CSIE, 2012.

[34] M. Liu, Y. Liu, and Q. Yang. Predicting best answerers for new questions in community question answering. In *11th International Conference on Web-age InformationManagement,*, pages 127–138. Springer-Verlag, 2010.

[35] X. Liu, W. B. Croft, and M. B. Koll. Finding experts in community-based question-answering services. In *14th ACM Conference on Information and Knowledge Management*, pages 315–316. ACM, 2005.

[36] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro.

[37] A. Mockus and J. D. Herbsleb. Expertise browser: A quantitaive approach to identifying expertise. In *International Conference on Software Engineering (ICSE 2002)*, pages 503–512. ICSE, 2002.

[38] A. Pal, S. Chang, and J. A. Konstan. Evolution and experts in question answering communities. In *ICWM '12*. AAAI, 2012.

[39] J. Pouwelse, p. Garbacki, D. Epema, and H. Sips. Pirates ans samaritans: A decade of measurements on peer production and their implications for net neutrality and copyright. *Telecommunications Policy*, 32(11):701–712, 2008.

[40] M. Qu, G. Qiu, C. Zhang, H. Wu, J. Bu, and C. Chen. Probabilistic question recommendation for question answering communities. In *18th International Conference on World Wide web, WWW '09*, pages 1229–1230. ACM, 2009.

[41] Badrul. Sarwar, George. Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *10th International Conference on World Wide Web, WWW '01, New York, NY, USA*, pages 285–295. ACM, 2001.

[42] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. In *25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '02, New York, NY, USA*, pages 253–260. ACM, 2002.

[43] Yang J. Bozzon A. Tagarelli A. Silvestri, G. Linking accounts across social networks: the case of stackoverflow, github and twitter. In *1st International Workshop on Knowledge Discovery on the WEB*, 2015.

[44] Automattic Team. A globally recognized avatar.

[45] Automattic Team. Image requests.

[46] Viet-Trung TRAN. Recommender systems: Content-based and collaborative filtering, 2015.

[47] B. Vasilescu, V. Filkov, and A. Serebrenik. Stackoverflow and github: associations between software development and xrowdsourced knowledge. In *Social Computing (SocialCom), 2013 International Conference*, pages 188–195. IEEE, 2013.

[48] B. N. Waber and A. Pentland. Recognizing expertise. In *Winter Conference on Business Intelligence, University of Utah, Utah, USA, 2009*, 2009.

[49] Jie Yang, Giuseppe Silvestri, Andrea Tagarelli, Alessandro Bozzon, and Gert-Jan Houben. Cross-platform expertise characterisation in online software development communities.

[50] Jie Yang, Ke Tao, Alessandro Bozzon, and Geert-Jan Houben. Sparrows and owls: Characterisation of expert behaviour in stackoverflow. In *International Conference on User Modeling, Adaptation, and Personalization*, pages 266–277. UMAP, 2014.

[51] Jun Zhang, Mark Ackerman, and Lada Adamic. Expertise networks in online communities: Structure and algorithms. WWW, 2007.

Appendix A

Glossary

In this appendix we give an overview of frequently used terms and abbreviations.

CQA: Community Question Answering

MEC: Mean Expertise Contribution

QA: Question Answer

TF-IDF: Term Frequency-Inverse Document Frequency

CF: Collaborative Filtering

CBF: Content-based Filtering

YA: Yahoo Answers

SO: Stack-overflow

SVM: Support Vector Machines

LFM: Latent Factor Model

VSM: Vector Space Model

UE: Ubiquitous Expertise

SpE: Specialist Expertise

SoE: Social Expertise

MEC: Mean Expertise Contribution

FM: Factorisation Machine

RMSE: Root Mean Square Error

NDCG: Normalised Discounted Cumulative Gain