A Computational Framework for Question Processing in Community Question Answering Services

LI, Baichuan

A Thesis Submitted in Partial Fulfilment
of the Requirements for the Degree of
Doctor of Philosophy
in
Computer Science and Engineering

The Chinese University of Hong Kong
February 2014

Thesis/Assessment Committee

Professor FU Wai Chee Ada (Chair)

Professor LYU Rung Tsong Michael (Thesis Supervisor)

Professor KING Kuo Chin Irwin (Thesis Supervisor)

Professor LEE Ho Man (Committee Member)

Professor ZHU Xiaoyan (External Examiner)

Abstract of thesis entitled:

A Computational Framework for Question Processing in Community Question Answering Services
Submitted by LI, Baichuan
for the degree of Doctor of Philosophy
at The Chinese University of Hong Kong in February 2014

Community Question Answering (CQA) services, such as Yahoo! Answers and Baidu Zhidao, provide a platform for a great number of users to ask and answer for their own needs. In recent years, the efficiency of CQA services for question solving and knowledge learning, however, is challenged by a sharp increase of questions raised in the communities. To facilitate answerers access to proper questions and help askers get information more efficiently, in this thesis we propose a computational framework for question processing in CQA services.

The framework consists of three components: popularity analysis and prediction, routing, and structuralization. The first component analyzes the factors affecting question popularity, and observes that the interaction of users and topics leads to the difference of question popularity. Based on the findings, we propose a mutual reinforcement-based label propagation algorithm to predict question popularity using features of question texts and asker profiles. Empirical results demonstrate that our algorithm is more effective in distinguishing high-popularity questions from low-popularity ones than other state-of-the-art baselines.

The second component aims to route new questions to potential answerers in CQA services. The proposed question routing (QR) framework considers both answerer expertise and answerer avail-

ability. To estimate answerer expertise, we propose three models. The first one is derived from the query likelihood language model, and the latter two models utilize the answer quality to refine the first model. To estimate answerer availability, we employ an autoregressive model. Experimental results demonstrate that leveraging answer quality can greatly improve the performance of QR. In addition, utilizing similar answerers' answer quality on similar questions provides more accurate expertise estimation and thus gives better QR performance. Moreover, answerer availability estimation further boosts the performance of QR.

Expertise estimation plays a key role in QR. However, current approaches employ full profiles to estimate all answerers' expertise, which is ineffective and time-consuming. To address this problem, we construct category-answerer indexes for filtering irrelevant answerers and develop category-sensitive language models for estimating answerer expertise. Experimental results show that: first, category-answerer indexes produce a much shorter list of relevant answerers to be routed, with computational costs substantially reduced; second, category-sensitive language models obtain more accurate expertise estimation relative to state-of-the-art baselines.

In the third component, we propose a novel hierarchical entity-based approach to structuralize questions in CQA services. Traditional list-based organization of questions is not effective for content browsing and knowledge learning due to large volume of documents. To address this problem, we utilize a large-scale entity repository, and construct a three-step framework to structuralize questions in "cluster entity trees (CETs)". Experimental results show the effectiveness of the framework in constructing CET. We further evaluate the performance of CET on knowledge organization from both user and system aspects. From a user aspect, our user study demonstrates that, with CET-based organization, users perform significantly better in knowledge learning than using list-based approach. From a system aspect, CET substantially boosts the per-

formance on question search through re-ranking.

In summary, this thesis contributes both a conceptual framework and an empirical foundation to question processing in CQA services.

論文題目 ： 一個社區問答服務中問題處理的計算框架

作者 ： 李百川

學校 ： 香港中文大學

學系 ： 計算器科學及工程學系

修讀學位 ： 哲學博士

摘要 ：

社區問答服務如雅虎知識+和百度知道爲大規模的使用者提供按需問答服務。近年來，隨著社區內大量增加的問題，社區問答服務在問題解決和知識學習的效率上面臨著不小的挑戰。爲了方便回答者找到合適的問題，幫助提問者更高效的獲取資訊，本論文提出了一個社區問答服務中問題處理的計算框架。

該計算框架中包含三部分：流行度分析與預測，路由，以及結構化。第一部分分析了影響問題流行度的因素，發現使用者和話題的交集造成了不同的問題流行度。基於這個發現，我們提出了一個基於相互增強的標籤傳播演算法，利用問題文本和提問者簡介預測問題流行度。實驗結果證明提出的演算法比先進的基準方法更能區別高流行度和低流行度的問題。

第二部分目的在於把新提出的問題路由給潛在的回答者。我們提出的問題路由框架考慮了回答者的專業知識和可用性。爲了估算回答者的專業知識，我們提出了三個模型。第一個模型來源於查詢詞相似語言模型，之後的兩個模型通過加入回答品質進一步優化第一個模型。對於估計回答者的可用性，我們借助了一個自回歸模型。實現結果證明引入答案品質顯著提高了問題路由的效果。此外，利用相似回答者在相似問題上的答案品質可以做出更準確的回答者專業知識預測，進而提高路由性能。回答者的可用性估計則進一步提高了路由的效果。

在問題路由中，回答者專業知識估計起到了至關重要的作用。然而目前的方法使用全部的簡介去對所有回答者進行估計，效率不高又費時。爲了解決這個問題，我們借助問題所在的類別構建了類別-回答者索引來過濾不相關的回答者，並提出了類別敏感的語言模型來估計使用者專業知識。實驗結果說明了：一，類別-回答者索引極大縮小了相關回答者的範圍，降低了計算時間；二，類別敏感的語言模型相比現今的基準方法，可更準確估計回答者專業知識。

在框架的第三個部分，我們提出了一個新穎的基於分層實體的方法結構化社區問答服務中的問題。由於大量文檔的存在，傳統的基於清單的問題組織在內容流覽和知識學習上效率低下。爲了解決這個問題，我們利用大規模的實體庫，構建了一個三步框架把問題結構化到“實體樹”中。實驗結果反映了該框架的有效性。我們進一步從使用者和系統兩方面評價實體樹在組織知識上的表現。在用戶層面上，用戶調查表明，使用者在基於實體樹的問題組織上知識學習的表現比基於列表的顯著提高。在系統層面上，實體樹通過再排序明顯提高了系統的問題搜索效果。

概括起來，該論文在概念框架和實證基礎兩方面爲社區問答中的問題處理做出了貢獻。

Acknowledgement

I would like to express my sincere gratitude and appreciation to my supervisors, Prof. Irwin King and Prof. Michael R. Lyu, as well as my thesis committee members Prof. Jimmy Lee, Prof. Ada Fu and Prof. Xiaoyan Zhu, who provided me with constant support and critical advice for my thesis. I am indebted to Prof. Helen Meng, Prof. Baley Mak, and Prof. Jeffrey Xu Yu from The Chinese University of Hong Kong, Prof. Yuncai Liu, Prof. Yan Jin, and Prof. Xubo Yang from Shanghai Jiao Tong University, Prof. Weizu Huang and Prof. Wen Zhao from Northeastern University and other professors and experts, who provided great help in the research.

I would like to thank my mentors, Dr. Edward Y. Chang and Dr. Xiance Si in Google, and Dr. Chin-Yew Lin in Microsoft Research Asia, for their kind guidance, consistent support, and insightful suggestions when I was visiting as a research intern. I also would like to thank Fangtao Li, Zhiyuan Liu, Jing Liu, Zhen Liao, Zhicheng Zheng, Kai Wang, Decheng Dai, Yu Qiao, Maxis Kao and many others for their discussions in my internships.

I thank Hao Ma, Haiqin Yang, Tom Chao Zhou, and Zibin Zheng for their efforts and constructive discussions in conducting the research work in this thesis. I also thank my colleagues in the web intelligence and social computing group, Xin Xin, Wei Wang, Yi Zhu, Mingzhen Mo, Dingyan Wang, Guang Ling, Shenglin Zhao, Hongyi Zhang, Priyanka Garg, Shouyuan Chen, Chen Cheng, Xixian Chen, Junjie Hu, and Tong Zhao.

I would like to thank Tan Jin, Ronghua Li, Renxin Mao, Jinlong

Tu, Xinyu Chen, Yangfan Zhou, Junjie Xiong, Yilei Zhang, Qirun Zhang, Jieming Zhu, Ning Wang and many others.

Last but not least, I want to thank my parents. Without their deep love and great support, this thesis would never have been completed.

To my beloved parents.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Since the inception of forums for asking and answering questions, Community Question Answering (CQA) services have been providing users with web platforms to obtain useful information, such as Yahoo! Answers[1], Baidu Zhidao[2], and Quora[3]. For instance, lunched on 2005, Yahoo! Answers has become the most popular CQA portal in the world, with 300 million questions asked as of June 2012[4]. Recently, a novel CQA site Quora [115, 142], which integrates a social network into its structure, has grown dramatically. Between June 2011 and June 2012, it saw a 350 percent growth in terms of total unique visitors. By June 2012, Quora has garnered over 1.5 million unique visitors per month[5] despite its short history (Quora was launched in June 2009).

CQA services are alternatives to question answering (QA) systems [147, 49, 53] and web search engines [15, 16]. QA aims to build systems that can automatically answer questions in forms of

[1]http://answers.yahoo.com/

[2]http://zhidao.baidu.com/

[3]http://www.quora.com/

[4]http://searchengineland.com/yahoo-answers-hits-300-million-questions-but-qa-activity-is-declining-127314/

[5]http://www.digitaltrends.com/social-media/silently-but-surely-quora-is-growing-rapidly-thanks-to-google/

Table 1.1: Comparison between CQA and QA.

Aspect	CQA	QA
Answer provider	registered users	machine
Answer speed	unsure	usually several seconds
Answer quality	varied	high for objective questions
Question search	support	usually not support
Question browsing	support	usually not support
Question organization	semi-structured category hierarchies or folksonomies	usually store questions in structured knowledge base

natural languages. Traditional QA systems usually utilize structured or unstructured databases to construct answers of *restricted-domain* questions [27, 104] or *open-domain* questions [119, 103]. However, an ideal *open-domain* QA system is difficult to achieve due to the varieties of question types and topics, the large-scale background knowledge, etc. With the development of the Internet, web search engines are more widely accepted than QA systems due to their prompt and precise searching results, although they merely provide relevant documents rather than answers. Search engines like Google[6] and Bing[7] are able to deal with simple and objective queries, but cannot answer complicated and subjective information needs [81], like seeking for opinions and recommendations.

CQA systems thus aim to solve subjective, specific, and open-ended questions [89, 128]. Different from traditional QA systems, CQA systems allow users to answer questions of other users (see the comparison between CQA and QA in Table 1.1).

In recent years, the efficiency of CQA services, however, is challenged by a sharp increase of questions raised in the communities.

[6]http://www.google.com/
[7]http://www.bing.com/

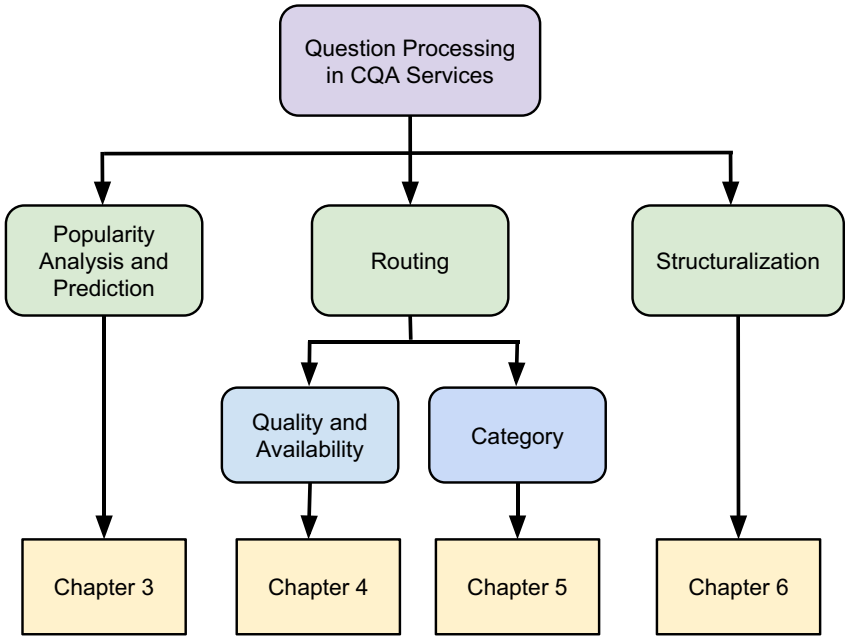

Figure 1.1: The computational framework for question processing in CQA services.

Such increasing amount of questions have thus influenced access of answerers to their appropriate questions, with the process of question answering being hindered [41]. Moreover, questions in CQA services are organized through a list structure under a category hierarchy [100, 156], which is ineffective for information browsing and knowledge learning [71]. To facilitate answerers access to proper questions and help askers obtain information more effectively, we propose a computational framework for question processing in CQA services.

Figure 1.1 shows the framework, which consists of three components: popularity analysis and prediction, routing, and structuralization. As the key part of the framework, question routing [69] refers to routing newly posted questions to potential answerers; the appropriateness of potential answerers is estimated based on archives of their previously answered questions. By routing, new questions

are directly presented to answerers who are available and have corresponding expertise, and the wait time of askers is reduced. On the other hand, answerers no longer need to browse and find questions to answer, but receive recommended questions. Therefore, the efficiency of QA is improved. Before routing, the framework first analyzes the factors affecting question popularity (so called "social quality" in [68], QP hereafter), and predicts whether a question would receive answers regarding to both quantity and speed. After routing, questions obtain answers more efficiently and become solved. Solved questions are valuable for future users to seek information. To facilitate users on browsing questions and learning knowledge, this framework then includes a question structuralization component, which structuralizes questions (and their answers) with a hierarchical entity-based approach.

Under the question processing framework, we first conduct two studies to investigate QP in CQA services. In the first study, we apply statistical analysis to find the factors influencing QP. In the second study, we propose a mutual reinforcement-based label propagation algorithm to predict QP using features of question texts and asker profiles.

To promote the efficiency of question answering, in the second work we propose a question routing (QR) framework to reduce askers' wait time and improve user satisfaction. The concept of QR contains two meanings: (1) questions are routed to the "right" answerers who can provide high quality answers; and (2) answerers who receive the routed question must be able to provide quick response.

The whole process of QR is as follows. For a question to be routed, we first extract all answerers in the portal and build their answering performance profiles. Then we estimate each candidate's expertise on the routed question based on his performance profile. Meanwhile, the availability of answerers is estimated based on recent activities. Finally, we rank all the answerers based on both expertise and availability.

The essence of QR rests on the estimation of answerer expertise (expertise estimation, hereafter) in routing questions. Volumes of studies have been conducted regarding expertise estimation, including the Query Likelihood Language Model (QLLM) [84], the Cluster-based Language Model (CBLM) [167], and a mixture of Latent Dirichlet Allocation (LDA) [10] and QLLM [80]. However, two problems of applying those models to expertise estimation are noted:

- Irrelevant answerers: For *all answerers*, the expertise has been estimated for QR, even to answerers without any experience of routed questions;
- Irrelevant questions: For *an answerer*, a complete set of questions the answerer has answered is utilized in the models, although a certain amount of answered questions might be irrelevant to questions to be routed.

As a result, the irrelevant answerers and questions would increase computational costs, rather than contribute to accuracies of expertise estimation.

In CQA services, askers must choose a category for the question they asked. Categories would allow much latitude in filtering irrelevant answerers among *all answerers*, together with screening irrelevant questions of *an answerer* to enhance the efficiency of expertise estimation. Cao et al. [19, 21] confirmed a significant improvement of question retrieval due to incorporating question category in various retrieval models (i.e., Vector Space Model (VSM) [126], BM25 [123], QLLM [84], Translation Model [56], and Translation-based LM [150]). To date, few attempts, however, have been made regarding category information in studies of QR.

The third work is designed to fill the gap. In this work, we utilize question categories to filter irrelevant answerers among *all answerers* and irrelevant questions in profile of *an answerer* for expertise estimation. More specifically, it constructs category-answerer

indexes for filtering irrelevant answerers, and develops category-sensitive LMs for estimating answerer expertise. By incorporating question category, an improvement in efficiency of routing questions is expected including two aspects:

- Higher accuracies: category-sensitive LMs will produce more accurate results of expertise estimation for routing questions as only relevant questions in profile of *an answerer* are employed.
- Lower costs: computational costs for routing questions will be reduced due to the shortened list of answerers with purified answer profiles used for expertise estimation.

At present, questions in CQA services are organized in a list structure with extra information (e.g., category hierarchies in Yahoo! Answers and social tags in Quora). This "list-of-content" (list-based approach) is simple and straightforward, but ineffective for browsing and knowledge learning. For example, a user who plans a trip to Hong Kong would like to discover all aspects of this city, like transportation, accommodation, food, etc. In this scenario, he may browse some relevant categories like "Travel:Hong Kong" to get useful information. He may also issue a query like "travel in Hong Kong" to search relevant questions. However, both browsing and searching provide the user a list of relevant contents, not the direct knowledge. Thus, the user must read these contents, understand them, classify them into various topics, and gain valuable knowledge himself. Obviously, it is ineffective and time-consuming.

The above problem calls for a new approach to structuralize questions, which allows users to browse information and seek knowledge more effectively. Traditionally, we can utilize topic models [10] or social tagging [43] to structuralize questions. However, for topic models, it is not easy to control the granularity of topics, and it is hard for users to interpret a topic only based on the multinomial distribution [98]. For social tagging, it is not applicable in many sites and has the sparsity problem [132]. Thus, both topic models and

social tagging are not suitable for structuralizing questions in CQA services.

To address the above issues, in the fourth work we propose a novel hierarchical entity-based approach, known as "cluster entity tree" or CET, to structuralize questions in CQA services by leveraging an existing large-scale entity repository. A CET is a tree structure based on entity co-occurrence. In addition, topic-coherent questions are grouped to the same cluster on each layer of a CET. In this work, we propose a three-step framework to construct CETs from question texts, namely entity extraction, tree construction, and hierarchical entity clustering. By utilizing a large-scale entity repository, CET avoids the granularity, interpretation, and sparsity problems. Entity repositories like Freebase[8] provide a large number of named entities across various pre-defined topics, which avoid the granularity and sparsity problems. In addition, they usually give descriptions of entities, which prevent the interpretation problem.

Empirical results validate the effectiveness of the framework in constructing CET. We further evaluate the performance of CET on knowledge organization from both user and system aspects. Our user study demonstrates that, with CET-based organization, users perform significantly better in knowledge learning than using the list-based approach. In addition, CET substantially boosts system performance on question retrieval through re-ranking.

1.2 Thesis Contribution

The main contributions of this thesis are described as follows:

1. **Analyzing and Predicting Question Popularity**
   We conduct the first study on analyzing the factors of QP and observe that the interaction between askers and topics results in the differences of QP. Based on our observations, we fur-

[8] http://www.freebase.com/

ther propose a Mutual Reinforcement-based Label Propagation (MRLP) algorithm to predict QP. Our experimental results demonstrate the effectiveness of our algorithm in distinguishing high-popularity questions from low-popularity ones.

2. **Question Routing**
   We introduce the concept of QR in CQA services, which routes new questions to appropriate answerers who are most likely to provide answers in a short period of time. Our contributions include: (1) Designing a QR framework which considers both answerer expertise and answerer availability; (2) Proposing three models to estimate answerer expertise under the framework; and (3) Demonstrating the effectiveness of our QR framework through empirical studies.

3. **Routing with Category Information**
   We propose a category-sensitive approach for QR. The incorporation of question category is designed to involve filtering irrelevant answerers and estimating answerer expertise for routing questions to potential answerers. For filtering irrelevant answerers, we construct category-answerer indexes; for estimating answerer expertise, we develop category-sensitive LMs. We conduct intensive experiments, and results demonstrate that higher accuracies of routing questions with lower computational costs are achieved, relative to traditional query likelihood language model as well as state-of-the-art baselines.

4. **Questions Structuralization with Hierarchical Entity Tree**
   We propose a hierarchical entity-based approach for structuralizing questions in CQA services. By using a large-scale entity repository, we design a three-step framework to organize questions in a novel hierarchical structure called “cluster entity tree (CET)”. With Yahoo! Answers as a test case, we conduct experiments and the results show the effectiveness of our framework in constructing CET. We further evaluate the performance of

CET on question structuralization in both user and system aspects. From a user aspect, our user study demonstrates that, with CET-based structure, users perform significantly better in knowledge learning than using traditional list-based approach. From a system aspect, CET substantially boosts the performance of traditional information retrieval models.

1.3 Thesis Organization

The rest of this thesis is organized as follows:

- **Chapter 2**
  In this chapter, we first review some background knowledge of CQA services. We then present related work in question processing and answer processing.

- **Chapter 3**
  In this chapter, we analyze the factors affecting QP and propose a graph-based approach to predict QP, which involves three dimensions: (1) user attention; (2) answering attempt; and (3) best answer. We first present the experimental data and the ground truth setting. Two studies are then described. In the first study, we examine the effects of askers and topics on QP. We observe that topics themselves cannot determine QP, and the interaction between askers and topics is mostly related to QP. This observation motivates us to design a novel algorithm to predict QP. In the second study, we propose a graph-based algorithm called "Mutual Reinforcement Label Propagation" (MRLP) for QP prediction. Because MRLP is an iterative system, we further prove the convergence of MRLP. Empirical studies on Yahoo! Answers data demonstrate the superiority of MRLP over other state-of-the-art methods. This chapter is based on our work in [68].

- **Chapter 4**
  In this chapter, we propose a question routing framework to recommend new questions to potential answerers. The framework consists of four components: (1) performance profiling; (2) expertise estimation; (3) availability estimation; and (4) answerer ranking. For a new question, we first identify all answerers and build their answering performance profiles. Then, we estimate each candidate's expertise on the routed question based on his performance profile. We estimate answerer expertise from two aspects: with and without answer quality. When incorporating answer quality, we propose two models. The *Basic Model* assumes the answerer's answer quality on a new question is the weighted average of similar questions' answer quality he answered previously. Meanwhile, the *Smoothed Model* further utilizes similar answerers to smooth answer quality estimation. During the question-user matching, we propose an autoregressive model to estimate answerer availability. Finally, all answerers are ranked according to their expertise scores and availability scores. We conduct experiments with Yahoo! Answers data set, and the results demonstrate the effectiveness of our framework. In addition, utilizing similar answerers' answer quality on similar questions provides a more accurate expertise estimation, and obtains better QR performance. Furthermore, availability estimation boosts the performance of QR. This chapter is based on our work in [69].

- **Chapter 5**
  This chapter focuses on incorporating category information in QR. Traditional question routing approaches usually take the complete set of answerers as candidates, even many of them are not familiar with routed questions. In addition, these methods utilize all answered questions as user profiles, although a certain amount of answered questions might be irrelevant to questions to be routed. To solve this problem, we utilize cate-

gory information provided in CQA services to sift out irrelevant answerers, and make better expertise estimation. To filter irrelevant answerers, we build two category-answerer indexes. To better estimate answerer expertise, we propose two category-sensitive language models. The first model utilizes the routed question' category to construct language models within that particular category. The second model incorporates profiles of similar categories to solve the sparsity problem and improve estimation accuracy. Experiments with large-scale data sets provide empirical evidence to validate the application of category to QR. This chapter is based on our work in [70].

- **Chapter 6**
  In this chapter, we propose a novel hierarchical entity-based approach, known as "cluster entity tree (CET)", to structuralize questions by leveraging an existing large-scale entity repository. To construct CETs, we design a three-step framework: entity extraction, tree construction, and hierarchical entity clustering. Then, we conduct a user study to investigate the influence of CET on question browsing and knowledge learning. In addition, we extend CET to question retrieval, and propose a CET-based question re-ranking algorithm. We perform re-ranking with Yahoo! Answers data and report that CET-based re-ranking substantially improves the performance of both the VSM and the QLLM. This chapter is based on our work in [71].

- **Chapter 7**
  The last chapter summarizes this thesis and raises some future directions that can be further explored.

□ **End of chapter.**

Chapter 2

Background Review

2.1 Yahoo! Answers

Launched on 2005, Yahoo! Answers has become the most popular CQA portal among the world, with 300 million questions asked as of June 2012. On average, two questions are asked and six are answered every second.

In Yahoo! Answers, questions are organized through a category hierarchy, which consists of 26 top categories and over 1,000 leaf categories. These categories provide shadow semantics to the corresponding questions and answers in these categories, and facilitate question browsing and searching. When a user asks a question, he has to select the specific leaf category that the question belongs to.

The life of a question in Yahoo! Answers is shown in Figure 2.1. When a new question is posted, its status becomes *Open* and this status will remain for four days. The time period can be extended by the asker upon expiration. During the *Open* period, other users can post their answers to this question. If no answer is obtained, it will expire and be automatically deleted. If the question receives at least one answer, the asker is allowed to select one answer as the best answer, starting from one hour after obtaining the first answer, or he can ask the community to vote for best answer. In the first case, the question life ends with changing to the *Resolved* status. In the second case, however, the question's status turns to *In Voting*, which

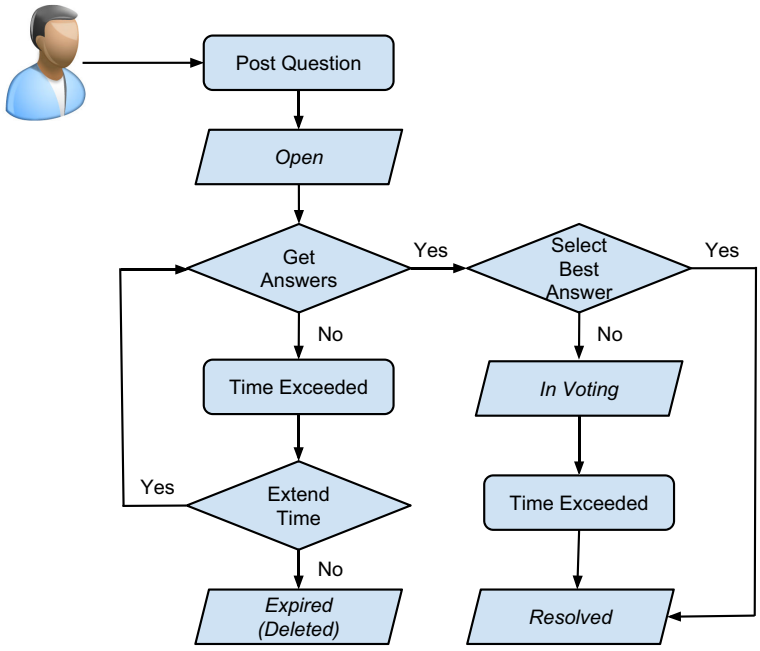

Figure 2.1: A question's life in Yahoo! Answers.

means all registered users have the right to vote for best answer. At the end of this phrase the answer with most votes would be selected as best answer, and the question's status would consequently becomes *Resolved*.

2.2 Studies in CQA Services

Most studies of CQA services focus on question and answering processing, which aim to achieve effective and efficient QA. As shown in Figure 2.2, these studies include:

1. **Question Retrieval.** It finds semantically equivalent questions to users' new questions (queries) [56, 163, 121]. Question retrieval saves users' time if their questions have been asked and well solved.

2. **Question Classification.** It classifies questions based on vari-

ous properties, such as urgency [90], subjectivity [73, 72, 166], quality [63, 30, 28], and topics [23]. Question classification can be utilized for improving question retrieval. It also helps to understand user intent, which enables a CQA system to provide better services.

3. **Question Routing.** It routes new questions to appropriate answerers [58, 112, 148]. In some literatures [139], it is also called question recommendation. In CQA services, questions are passively listed, and may not be seen by potential answerers. Question routing seeks potential answerers actively when new questions are posted, which makes QA more effective.

4. **Answer Quality Evaluation.** It analyzes the features that related to answer quality, and utilizes them to predict answer quality [140, 42, 26]. Sometimes, predicting the best answer is the main propose of this study [11, 80]. As any registered users can answer any open questions, the quality of answers varies. Answer quality evaluation helps both the system and the askers to judge answer quality and select the best answer.

5. **Answer Summarization.** It summarizes all received answers to a complete, structured, succinct, and quality answer [85, 24, 113]. For some subjective questions such as seeking for opinions or asking for recommendations, answerers may provide various answers. It would be inefficient for askers to look through answers if the questions obtain many answers. Therefore, categorizing and summarizing answers will help askers get information more effectively.

In addition, there are user-based research topics which focus on expert finding [160, 92], user analysis [44, 1], and user satisfaction prediction [87, 88, 81]. Common approaches for expert finding include language models [84], link analysis [59, 109], Bayesian information criterion [14], competition-based models [78], graph-based

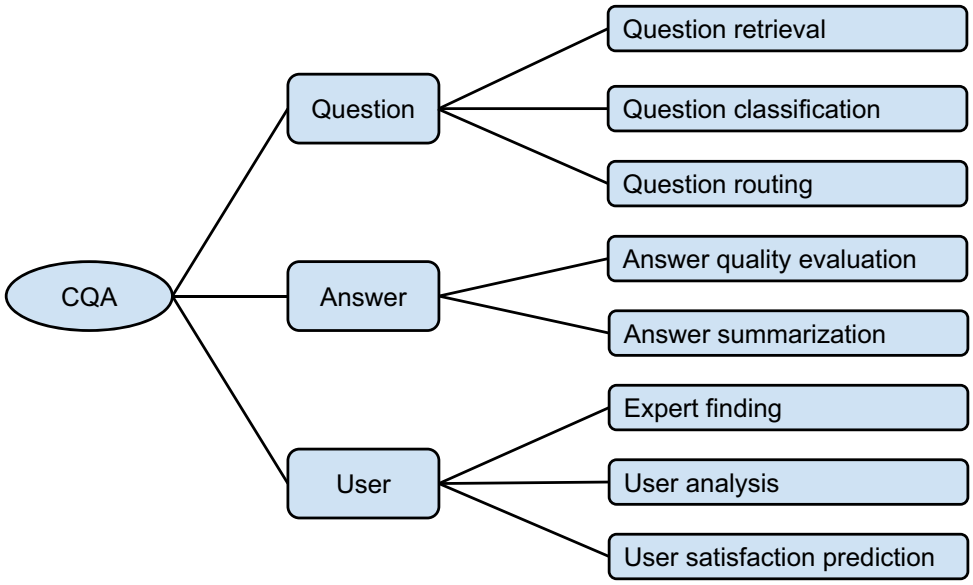

Figure 2.2: Research topics of CQA services

models [153, 17], Gaussian classification models [111], and hybrid approaches [60, 149, 161]. Studies on user analysis include user classification [39], collaboration analysis [40], together with activity and behavior analysis [124, 99]. Moreover, some studies focus on system level analysis, such as community [86, 106, 74, 75], system architecture [134, 51, 108], reputation system [105], sentiment [65], and information quality [35].

In the following, we review related work on the listed five topics in turn. Although user-based research is not explicitly covered, some issues such as expert finding and user analysis are implicitly contained in the above topics. For instance, question routing includes expertise estimation [69, 70], and some answer quality evaluation approaches utilize user analysis to enhance prediction [146, 42].

Table 2.1: Basic models for question retrieval.

Model	Paper
Vector space model	[32, 19]
Okapi BM25 model	[56, 19]
Query likelihood language model	[56, 32, 21, 19]
Translation model	[32, 150, 19]
Translation-based language model	[150, 19]

2.3 Question Retrieval

In CQA services, question retrieval means finding semantically equivalent questions to users' new queries. Here queries can be either keywords or natural questions. Various information retrieval models have been applied on question retrieval, as summarized in Table 2.1. In the following, we first briefly introduce these basic models (Section 2.3.1). Then, we summarize some advanced approaches which incorporate category information (Section 2.3.2), syntactic and semantic information (Section 2.3.3), question segmentation (Section 2.3.4), and answer quality (Section 2.3.5) for retrieving similar questions. For consistency, we use the notations listed in Table 2.2 throughout this section.

2.3.1 Basic Models

Vector Space Model (VSM). The vector space model calculates the cosine similarity between a query and a question, and has been widely applied for question retrieval [32, 19]. In one popular variation of this model [172], the similarity score of query q and question d is calculated as follows:

$$Sim_{_{VSM}}(q,d) = \frac{\sum_{t \in q \cap d} w_{q,t} w_{d,t}}{W_q W_d}, \tag{2.1}$$

Table 2.2: Notations used in Section 2.3.

Symbol	Description
q	a query
d	a question
a	an answer
t	a term
$Coll$	the whole collection of questions
N	the number of questions in the whole collection
df_t	the number of questions containing the term t
$tf_{t,q}$	the frequency of term t in q
$tf_{t,d}$	the frequency of term t in d
$tf_{t,Coll}$	the frequency of term t in the whole collection
$Sim(q,d)$	similarity score between a query q and a question d

where

$$w_{q,t} = \ln(1 + \frac{N}{df_t}), \tag{2.2}$$

$$w_{d,t} = 1 + \ln(tf_{t,d}), \tag{2.3}$$

$$W_q = \sqrt{\sum_t w_{q,t}^2}, \tag{2.4}$$

$$W_d = \sqrt{\sum_t w_{d,t}^2}. \tag{2.5}$$

Note that $w_{q,t}$ captures the IDF (inverse document frequency) of term t in the collection, and $w_{d,t}$ captures the TF (term frequency) of term t in d. Since W_q is a constant for a given query, it can be neglected without affecting the final rankings.

Okapi BM25 Model. One shortcoming of the vector space model is that it favors short questions. As shown in Eq. (2.1), short questions with small W_d will get higher similarity scores when other conditions are the same. To overcome this problem, the Okapi BM25

model takes into account question length [56]. The following presents a widely used version of Okapi BM25 model [172, 19].

$$Sim_{_{BM25}}(q,d) = \sum_{t \in q \cap d} w_{q,t} w_{d,t}, \tag{2.6}$$

where

$$w_{q,t} = \ln(\frac{N - df_t + 0.5}{df_t + 0.5}) \frac{(k_3+1)tf_{t,q}}{k_3 + tf_{t,q}}, \tag{2.7}$$

$$w_{d,t} = \frac{(k_1+1)tf_{t,d}}{K_d + tf_{t,d}}, \tag{2.8}$$

$$K_d = k_1(1 - b + b\frac{W_d}{W_A}). \tag{2.9}$$

In the above equations, k_1, b, and k_3 are parameters. W_d is the question length (number of words) of d, and W_A is the average question length in the whole collection. In this model, the influence of question length to similarity score is determined by the parameter b. If b is small, question length contributes little to similarity score, and vice versa.

Query Likelihood Language Model (QLLM). The basic idea is to estimate a language model for each question, and then rank questions by the likelihood of the query according to the estimated language models. With the Jelinek-Mercer smoothing [157], the language model is:

$$Sim_{_{QLLM}}(q,d) = \prod_{t \in q} (1-\lambda) P_{ml}(t|d) + \lambda P_{ml}(t|Coll), \tag{2.10}$$

where

$$P_{ml}(t|d) = \frac{tf_{t,d}}{\sum_{t' \in d} tf_{t',d}}, \tag{2.11}$$

$$P_{ml}(t|Coll) = \frac{tf_{t,Coll}}{\sum_{t' \in Coll} tf_{t',Coll}}. \tag{2.12}$$

In Eq. (2.10), $\lambda \in [0,1]$ is a smoothing parameter which adjusts the weight of whole collection's influence on the similarity score.

Translation Model (TM). QLLM may suffers when there are few vocabulary overlaps between q and d. For instance, "How to lose weight" and "I want to be slim, what should I do" are two semantically similar questions. However, they share few common words, which leads to ineffective similarity estimation for QLLM. The translation model [56, 150] exploits word-to-word translation probabilities in the language modeling framework, and alleviates the above vocabulary mismatch problem. In a typical translation model [56],

$$Sim_{_{TM}}(q,d) = \prod_{t \in q}(1-\lambda)\sum_{w \in d} T(t|w)P_{ml}(w|d) + \lambda P_{ml}(t|Coll), \tag{2.13}$$

where $T(t|w)$ denotes the probability that word w is the translation of word t. In [56], it assumes that the probability of self-translation is 1, i.e., $T(w|w) = 1$.

Translation-based Language Model (TBLM). Xue et al. [150] combined QLLM and TM for question retrieval, and showed that the model gained better performance than both QLLM and TM. In this model,

$$\begin{aligned} Sim_{_{TBLM}}(q,d) &= \prod_{t \in q}(1-\lambda)(\beta \sum_{w \in d} T(t|w)P_{ml}(w|d) \\ &+ (1-\beta)P_{ml}(t|d)) + \lambda P_{ml}(t|Coll), \end{aligned} \tag{2.14}$$

where β is the weighting parameter for QLLM and TM. When $\beta = 0$, TBLM becomes QLLM; when $\beta = 1$, TBLM reduces to TM.

2.3.2 Category Information

CQA services like Yahoo! Answers usually provide a category hierarchy to organize questions in different levels of abstraction. Category information provides extra knowledge for question retrieval. Intuitively, the more related a category is to a query q, the more likely that the category contains questions relevant to q. Utilizing

category information, Cao et al. [19] proposed a series of retrieval models based on the basic models in Table 2.1. The basic idea of these models is to combine two similarity scores in question retrieval: one is a global relevance score (denoted as $S_{q,cat(d)}$) which models the similarity between q the d's category ($cat(d)$), and the other is a local relevance score (denoted as $S_{q,d}$) between q and d within d's category. After normalizing both the global relevance and the local relevance scores, the final similarity score is computed as

$$Sim_{_{CI}}(q,d) = (1-\beta)N(S_{q,d}) + \beta N(S_{q,cat(d)}), \tag{2.15}$$

where β is the weighting parameter and $N(\cdot)$ is the normalization function.

2.3.3 Syntactic Tree Matching

Bag-of-word approaches, such as VSM and QLLM, may perform poorly when similar questions do not share many common words. Under this circumstance, syntactic and semantic features become crucial. Wang et al. [144] employed syntactic features and proposed a syntactic tree matching (STM) approach for question retrieval.

Given a query q's parsing tree T_1 and a question d's parsing tree T_2, the similarity score of these two trees is defined as follows:

$$Sim_{_{STM}}(q,d) = \frac{S(T_1,T_2)}{\sqrt{S(T_1,T_1)S(T_2,T_2)}}, \tag{2.16}$$

$$S(T_1,T_2) = \sum_{r_1 \in T_1} \sum_{r_2 \in T_2} M(r_1,r_2), \tag{2.17}$$

where $M(r_1,r_2)$ is the matching score of two nodes r_1 and r_2. It is computed as follows:

$$M(r_1,r_2) = \begin{cases} \delta_{r_1}\delta_{r_2}\lambda^{S_{r_1}+S_{r_2}}\mu^{D_{r_1}+D_{r_1}}, & \text{if } r_1,\ r_2 \text{ are terminals;} \\ \delta_{r_1}^{\eta}\delta_{r_2}^{\eta}\lambda^{2\eta}\mu^{\eta[2-(1+nc(r_1))(D_{r_1}+D_{r_2})]\times G}, & \text{otherwise.} \end{cases} \tag{2.18}$$

In the above equation, δ_{r_1} denotes the importance of node r_1 in T_1, S_{r_1} and D_{r_1} represent the size and depth of the tree fragment with the root node r_1. S_{r_1} is defined by the number of nodes that the tree fragment contains, and D_{r_1} is defined as the level of the tree fragment root in the entire syntactic parsing tree. λ and μ are two parameters denoting the preference between size and depth, η is the total number of matched tree fragments, and $nc(r_1)$ means the total number of children of the node r_1. In addition,

$$G = \prod_{j=1}^{nc(r_1)} M(ch(n_1, j), ch(n_2, j)). \tag{2.19}$$

where $ch(n, j)$ is the j^{th} child of node n in the tree. To capture more semantic meanings, STM is further smoothed through utilizing WordNet [34].

2.3.4 Segmentation-aided Retrieval

A multi-sentence query (containing many sub-questions) usually contains different information needs, and the above retrieval models can hardly distinguish these different information needs and provide satisfying retrieval results. To address this, Wang et al. [145] proposed the segmentation-aided retrieval model to address this problem. This model contains two key steps: *question segmentation* and *question matching*.

- *Question segmentation* groups each sub-question with its context sentences and separates it from the other sub-questions;
- *Question matching* step matches two question sentences with the assistance of additional related contexts.

Segmentation-aided retrieval overcomes the drawback of bag-of-word retrieval models. However, since the model does not incorporate semantic information, it may suffer from the vocabulary mis-

match problem. In addition, question segmentation treats each sub-question equally important, and does not distinguish the importance of each information need.

2.3.5 Answer Quality

Bian et al. [8] employed the learning-to-rank technique [82] for question retrieval, which considers both question relevance and answer quality. In this framework, queries, questions, and answers are modeled as preference data, and gradient boosting [37] is applied for learning ranking function. Let $S = \{\langle x_i, y_i\rangle | x_i \succ y_i\}(i = 1, ..., N)$ denote the set of preference data, in which x and y are the feature vectors for two query-question-answer triples with the same query. $x \succ y$ means that x is preferred over y, i.e., x should be ranked higher than y.

Given S, the objective of learning to rank is to find a function h that minimizes the risk:

$$R(h) = \frac{1}{2}\sum_{i=1}^{N}(\max\{0, h(y_i) - h(x_i) + \tau\})^2, \tag{2.20}$$

where τ is a small-enough positive number. With gradient boosting, the ranking function h can easily be learned.

This approach is suitable for factoid questions where the preference data are easy to label. However, in CQA websites where there are many non-factoid questions, such as opinion questions and recommendation questions, for which answer quality is hard to measure. In this case, automatic extraction of preference data seems impossible and manual labeling is time-consuming.

2.3.6 Latent Semantic Tensor Indexing

The above retrieval models usually treat a question as a whole, but in the real world a question is composed of two components: question title and question content. Qiu et al. [121] represented questions

with a triple form $\langle$*question title*, *question content*, *answer content*$\rangle$, and proposed a latent semantic tensor indexing (LSTI) approach to model word association in different parts of triples, e.g., question title-question title and question title-question content. For a set of triples $\langle qt_i, qc_i, a_i \rangle$ $(i = 1, ..., N)$, a tensor $\mathcal{D} \in \mathbb{R}^{N \times 3 \times T}$ is utilized to represent the collection, where T is the number of terms. In $\mathcal{D}$, the three dimensions correspond to entries, parts, and terms respectively.

Given a new question q and an archived question d, their corresponding triples $\mathcal{D}_q \in \mathbb{R}^{1 \times 3 \times T}$ and $\mathcal{D}_d \in \mathbb{R}^{1 \times 3 \times T}$ are first projected to term space. Let $\hat{\mathcal{D}}_q$ and $\hat{\mathcal{D}}_d$ denote the projection matrices, the similarity score is then defined as the normalized Forensics inner product of these two matrices:

$$Sim_{_{LSTI}}(q, d) = \frac{\sum_{i,j} \hat{\mathcal{D}}_{q_{i,j}} \hat{\mathcal{D}}_{d_{i,j}}}{\sum_{i,j} \hat{\mathcal{D}}^2_{q_{i,j}} \times \sum_{i,j} \hat{\mathcal{D}}^2_{d_{i,j}}}. \tag{2.21}$$

LSTI is a unified model which solves the problem of semantic gap in question retrieval. However, tensor decomposition of CQA collections faces the sparsity problem in term space. In addition, due to the lack of answers, answer content contributes little for new questions.

2.4 Question Classification

Question classification aims to categorize questions into different groups based on various properties. Liu et al. [90] classified questions according to time-sensitivity. In their work, two classes are defined: urgent and non-urgent. Utilizing question text, category, and answer features, they applied both a SVM and a decision tree classifiers on a Yahoo! Answers data set. Ground truth was manually labeled on Amazon Mechanical Turk [1]. Empirical results show that the two classifiers performed similarly. In addition, question text

[1] https://www.mturk.com/mturk/welcome/

features alone provide satisfying classification results, while category features slightly improve the performance. However, answers are not helpful. As the first study to distinguish urgent questions from non-urgent ones, some limitations exist in [90]. First, it uses some regular expressions to find potentially urgent questions, which may lead to bias in their training data. Second, the information from asker profiles, which may provide extra help for analyzing questions' time-sensitivity, is not considered.

Li et al. [73] employed a SVM classifier to conduct binary classification using question and answer text features, such as character tri-grams, word uni-, bi-, and tri-grams. Due to the lack of training data and poorly-stated questions, the best accuracy is achieved at 0.74. To further improve the performance, they proposed a co-training [13] approach that exploits the association between questions and answers [72]. The intuition behind the co-training algorithm is that each question can be viewed in two aspects: question text and answer text. Based on the two views (two different feature sets), two separate classifiers were trained respectively. By automatically classifying the unlabeled examples, these two classifiers iteratively "teach" each other by giving their partners a newly "labeled" data, for which it can predict with high confidence. Experimental results show that the co-training approach improved the classification performance significantly. However, these two approaches only employ text features and ignored other helpful information, such as question categories and user profiles.

Harper et al. [46] addressed the above shortcomings. They analyzed the differences between conversational (objective) and informational (subjective) questions using human coding, statistical analysis, and machine learning algorithms. They first extracted questions from three CQA portals and asked annotators to label questions' subjectivity, writing quality, and archival value. They found that:

1. Humans could reliably distinguish between conversational and

informational questions in most cases.

2. The proportion of conversational questions differed at different portals.

3. On average, conversational questions received lower scores on both writing quality and archival value.

Based on these observations, they applied the sequential minimal optimization algorithm [117] to predict question subjectivity using category features, text features, and social network features. Empirical results indicate that category features lead to the best performance while text features perform worst. Using social network features yield slightly worse results than using category features. Their ensemble classifier achieves 89.7% classification accuracy across the three CQA portals, approaching human performance (91.4%). However, one drawback of this approach is that it requires manual labeling, which is time consuming when the data size is large.

To avoid manual labeling, Zhou et al. [166] recently investigated social signal features to automatically construct training data and predict question subjectivity in CQA services. Several heuristic features for question subjectivity identification, such as vote and source, were proposed. By utilizing these heuristic features, they automatically extracted training data with labels, and reported that the prediction results achieved 11.23% improvement over text features under the same experimental setting. However, their social signal features are available only when the questions have been posted for a while and have obtained answers, votes, likes, etc. As such, this approach cannot be utilized to handle newly asked questions.

2.5 Question Routing

In recent years, the efficiency of CQA services is been challenged by a sharp increase of questions in the communities. The increasing amount of questions have thus influenced access of answerers to

their appropriate questions, with the process of question answering being hindered [41]. To facilitate answerers access to proper questions, *Question Routing* (QR) has been initiated and developed in CQA services [29, 31].

The concept of QR refers to routing newly posted questions to appropriate users. From question aspect, QR looks for potential users who will provide good answers in time. From user aspect, QR recommends new and interesting questions to them. In CQA services, different approaches have been proposed for QR. To summarize, almost all of these methods involves three phases: *user profiling*, *question profiling*, and *matching*.

- *User profiling* refers to building up user profiles in terms of answering history of users.
- *Question profiling* means representing the information need of questions in appropriate forms.
- *Matching* represents computing the matching score between an user profile and a question profile.

After *matching*, answerers (or questions) are ranked according to the matching score. In the following, we review different approaches of QR in turn, and show how they handle these issues.

2.5.1 Topic Model-based QR

Guo et al. [41] proposed a QR approach which combined both term-level features and topic-level features. In their approach,

- *User profiling* integrates texts in questions and answers as user profiles. In addition, topic-level descriptions of user profiles are also exploited.
- *Question profiling* represents questions in both term-level and topic-level from question texts.

- *Matching* computes the similarity score from linear combination of term-level similarity (BM25) and topic-level similarity.

Qu et al. [122] utilized the probabilistic latent semantic analysis (PLSA) [50] for QR. In *user profiling*, a user-word aspect model was employed to model the joint probability of a user u and a word w:

$$P(u, w) = \sum_{z} P(u|z)P(w|z)P(z), \tag{2.22}$$

where $z \in \{z_1, ...z_k\}$ represents a topic. To estimate each probability, Expectation Maximization (EM) is employed to find the local optimal of the log likelihood of question collection. There is no *question profiling* in [122]. In the *matching* phrase, the joint probability of u and the new question q is estimated as the *matching* score:

$$P(u, q) = \prod_{i} P(u, w_i), \tag{2.23}$$

where w_i is the i^{th} word in q.

The above approaches have some limitations. First, answer quality is not considered in *user profiling*, and the impact of user availability is ignored in *matching*. Second, since these models do not employ external information, they cannot handle the cold start problem for new answerers.

2.5.2 Language Model-based QR

In language model-based QR [84, 167, 80], *user profiling* is conducted through estimating the probability of generating each word from answered questions, with maximum likelihood estimation (MLE) been used. *Question profiling* is omitted, and *matching* is based on QLLM described in Section 2.3.1.

Liu et al. [84] conceptualized expertise estimation as modeling text similarities between those of routed questions and those of answered questions in user profiles. As such, they utilized Query Likelihood Language Model (QLLM [102]), Relevance Language Model

(RLM [67]), and Cluster-based Language Model (CBLM [83]) in ranking answerer expertise, with QLLM performing the best. Liu et al. [84] proved feasibility of using language models for expertise estimation. However, all answerers' priors are treated equally in these language models, so the priors of each answerer to answer the routed question are identical. In real world, answerers who answered more questions should be assigned with higher priors.

Following this work, Zhou et al. [167] proposed three language models for *matching*.

1. Profile-based. The *matching* score is computed from QLLM, with all threads (QA pairs) in which the user provides answers as the document.

2. Thread-based. For each QA pair, the *matching* score is computed using QLLM, with both question and answer as the document. The final *matching* score is derived from the weighted summation of all *matching* scores for each thread.

3. Cluster-based. Threads with similar content are first clustered, and each cluster represents one topic that the user contributes to. Then, the *matching* score of each cluster is estimated from QLLM. Finally, the holistic *matching* score is the weighted summation of all clusters' *matching* scores.

After generating the rankings for new questions using the above language models, users are re-ranked based on their authority. A PageRank-based algorithm on the post-reply graph is employed to compute each user's authority.

One shortcoming of QLLM is that it suffers from the vocabulary mismatch problem. To address this concern, Liu et al. [80] combined LDA [10] with QLLM in *matching*. Besides, they utilized user authority and user availability as the user prior of QLLM. User authority is measured from the number of answers provided, and user availability is estimated based on the user's recent activity.

2.5.3 Classification-based QR

Zhou et al. [165] casted QR as a classification task, and employed a set of local and global features for *user profiling* and *question profiling*, which capture different aspects of users and questions. It is worth noting that some features are adopted to describe the semantic similarity between the question's language model and the user's language model (e.g., the KL-divergence between the current question's title/detail and all questions' title/detail the user has answered). From data analysis, they reported that question-user relationship features play a key role in identifying appropriate answerers. With these features and a set of training data, a one-class SVM classifier was constructed to predict whether a user would answer a question, and the prediction results were utilized in *matching*.

Dror et al. [29] proposed a multi-channel approach for assessing the match between a user and a question. In *question profiling*, a question is represented by three families of features: textual features, category features, and User IDs features. In *user profiling*, a user is profiled with seven channels of features: asker, best answerer, answerers, question voters, answer voters, best answer selectors, and question tracers. These seven channels are exactly those that are used for the social attribute annotating a question. Apart from that, they also allocated one more channel for explicit user attributes (e.g., a user can explicitly specify which keywords or topics he is interested in or which other users he prefers to follow). With question features and user features described above, they generated the interaction features. Several classifiers are utilized to predict whether a question matches a user in *matching*. Experimental results indicate that content features are more important than direct social relation features, and the gradient boosted decision tree (GBDT) algorithm obtains the best performance.

Table 2.3: Comparison among different approaches for question routing.

Approach	Paper	User Profiling	Question Profiling	Matching	Model Update	New User Problem	Super User Problem
Topic model	[41]	UQA	UQA	term-level similarity and topic-level similarity	all users and questions	yes	yes
	[122]	PLSA	PLSA	joint user and question probability over topics	all users and questions	yes	yes
Language model	[167]	word generation probability estimation	word generation probability estimation	language models	single user and single question	yes	yes
	[80]	word generation probability estimation and LDA	word generation probability estimation and LDA	language models (with user authority and user availability)	all users and single question	yes	yes
Classification-based model	[165]	feature extraction	feature extraction	SVM	single user and single question	yes	yes
	[29]	feature extraction	feature extraction	GBDT	single user and single question	yes	yes
DF model	[139]	three-level probability trees	LDA, lexical, and category features	vector similarity and proactive diversification	single user and single question	yes	no

2.5.4 Diversity and Freshness (DF model)

Szpektor et al. [139] found that only matching questions' information need with user profiles failed to capture user interests in a real system. In other words, "showing users just the main topics they had previously expressed interest in is simply too dull" [139]. However, adding a few topics slightly outside the core of their past activities may improve user engagement. With this intuition, they proposed a diversity- and freshness-aided QR approach.

In this approach, a question profile is represented by a LDA topic vector, a lexical vector, and a category vector. A user profile is represented as a probability tree, in which each node consists of a probability distribution that stands over various elements of the question model. A probability tree has three levels: top-category-distribution level, model-distribution level, and feature-distribution level. In the *matching* phase, both user profiles (probability trees) and questions profiles (feature vectors) are flattened to single vectors, and dot-product is employed for calculating the similarity score. To further improve the diversity and freshness of routed questions, they proposed a proactive diversification method.

2.5.5 Model Comparison

Table 2.3 compares different QR approaches from the following six aspects.

- *User profiling.* In topic model-based QR, different topic models are utilized to profile user expertise over topics. For language model-based QR, MLE is used to estimate word generation probability from user answered questions. Classification-based models extract various features from different aspects. The DF model builds a three-level probability tree for each user, representing user expertise over topics, words, and categories.

- *Question profiling.* The UQA model profiles questions together with users, while the PLSA model does not model questions. Language models employ MLE to estimate word generation probability from questions. Classification-based models extract various features from texts and contexts. The DF model builds three vectors for each question from topics models, texts, and question categories.

- *Matching.* The UQA model matches a question and a user from both term-level and topic-level similarities, and the PLSA model uses joint user-question probability over topics to calculate the matching score. *Matching* for language models is usually based on QLLM. In some models [80], user authority and availability are incorporated to improve matching. As for classification-based models, they train classifiers to predict whether the user would match the newly posted question. However, the DF model takes a ranking-based approach with proactive diversification.

- *Model updates.* When a new answer is submitted, topic models require re-profiling for all users since the topic distribution may be changed. Therefore, they are not suitable in online situation. Language models (except LDALM), classification-based models, and the DF model, however, only require updates for corresponding user. Among these models, most language models and the DF model are more suitable for online updates. As classification-based models may need re-training for the classifier, model updates would be time-consuming.

- *New user problem.* All approaches cannot deal with the cold start problem of new users due to lack of information. Some solutions in recommender system [2, 18, 131] may bring insight to address this issue, such as utilizing content data [127, 66], social tags [162, 33], social networks [97, 93, 94], etc.

- *Super user problem.* All approaches except the DF model face the super user problem. Through incorporating diversity and freshness, the DF model avoids routing too many similar questions to a super answerer. For other models, the more relevant questions a user has answered, the higher expertise the user would get, and the user thus obtains higher probability to receive routed questions. In future studies, load balancing techniques in networking [22] could be borrowed to address the problem.

2.6 Answer Quality Evaluation

Since CQA allows every user to answer any question, it is no exception that CQA services are filled with mixed-quality content. Some users could provide high quality answers which are relevant, concise, clear, helpful, and easy to understand. Meanwhile, others may type in their answers informally with error grammars and vocabulary. Some users even make fun of others by answering with nonsense or taking the opportunity for advertisement.

Therefore, evaluating answer quality becomes an essential issue in CQA services. Although the asker can examine answer quality himself and pick the best answer, it is usually time-consuming, especially when the number of answers becomes large. Automatically answer evaluation (AAE) addresses the problem. A good AAE system assists askers to choose the best answer and helps the service to distinguish high quality content from low ones. By utilizing various features that relate to answer quality, AAE directly predicts answer quality or conducts quality-based answer ranking. In the following, we first summarize the features investigated for AAE (Section 2.6.1), and then present different AAE algorithms (Section 2.6.2).

2.6.1 Features

Jeon et al. [57] observed that non-textual features could be utilized to estimate the quality of the document, and investigated 13 non-textual features for AAE. They report that, among these features, answer length and answerer expertise (e.g., category specialty, activity level, acceptance ratio, etc.) are the most salient features.

Following this work, Eugene et al. [3] investigated more features which may relate to answer quality. In their work, answer features, question features, user-user relationship features, question-answer relationship features and usage features are analyzed.

Shah and Pomerantz [129] further focused on two classes of features: quality criteria features [171] and QA features. All 13 quality criteria features[2] were first labeled by Mechanical Turk[3] workers, and these assessments were then matched with the actual asker's rating of a given answer. Empirical results show that the quality criteria they employed faithfully matches with asker's perception of a quality answer. Among quality criteria features, 'novel', 'original', and 'readable' have a significant impact on predicting the best answer. However, these features are usually subjective, and difficult to quantify. Therefore, they explored QA features which can be extracted from questions and answers automatically. It is reported that features extracted from questions exclusively do not help in AAE, while features extracted from answers exclusively achieve quite good performance. Different from the results of [57] and [3], they found 'the reciprocal rank of that answer' is the most distinguished feature. Moreover, they suggested that contextual information such as user profile was critical in evaluating and predicting answer quality.

[2] i.e., 'informative', 'polite', 'complete', 'readable', 'relevant', 'brief', 'conniving', 'detailed', 'original', 'objective', 'novel', 'helpful', and 'expert'.

[3] https://www.mturk.com/mturk/

2.6.2 Models

Approaches for automatically evaluating (or predicting) answer quality fall into two categories: classification-based and ranking-based. The first one treats this problem as a classification problem, extracts features to train classifiers, and predicts the class labels (e.g., good vs. bad) of new answers. The second category casts the problem as an information retrieval problem, in which answers are ranked according to their quality.

Classification-based models. Jeon et al. [57] built a maximum entropy based classifier with non-textual features to predict answer quality. First, non-textual features are extracted from questions, answers and answerer profiles. Second, since the maximum entropy model requires monotonic features (higher feature value represents stronger evidence), non-monotonic features are converted to probabilistic features using kernel density estimation (KDE) [55]. Finally, the probability of being a good answer or bad answer is:

$$p(y|x) = \frac{1}{Z(x)} \exp\left[\sum_{i=1}^{m} \lambda_i x_{f_i}\right], \tag{2.24}$$

where x represents question-answer pairs, y is the class label, m is the number of features, x_{f_i} denotes the i^{th} feature of x, $kde(\cdot)$ means the KDE function, and $Z(x)$ is a normalization factor. λ are parameters of the maximum entropy model, which can be estimated from [95]. In addition,

$$f_i(x, y) = \begin{cases} kde(x_{f_i}), & \text{if } f_i \text{ is non-monotonic;} \\ x_{f_i}, & \text{otherwise.} \end{cases} \tag{2.25}$$

Eugene et al. [3] extended this work and extracted more features (e.g., text and relation) to predict answer quality. Similar to [57], answer quality is divided into two classes. Several classification algorithms such as support vector machines and log-linear classifiers are tested in experiments, among which the stochastic gradient boosted trees [37] algorithm obtains the best performance with

AUC[4] 0.878. The relationship between question quality and answer quality is also investigated. It is observed that good answers are "much more likely to be written in response to good questions", while bad questions usually attract more bad answers. In addition to above models, Shah and Pomerantz [129] employed the logistic regression model to evaluate answer quality.

Ranking-based models. One problem of classification-based approaches is that questions and answers are modeled independently and their implicit semantic relationships are not incorporated. To better utilize QA relationships, ranking-based models are proposed.

Suryanto et al. [138] observed that authoritative users (experts) tend to provide high quality content. Based on this, an answerer expertise and answer relevance based approach is proposed to select good answers from a CQA system. The key idea is that the quality of an answer depends on the relevance degree to the question it answers as well as the answerer's expertise. Given a new question q, the overall score of an answer a is:

$$score(a) = rscore(q, a) \cdot qsocre_{\langle model \rangle}(q, a), \tag{2.26}$$

where $rscore(q, a)$ measures the relevance between q and a, which can be obtained through the query likelihood retrieval model. In addition, $qsocre(q, a)$ represents the quality of a, which is derived from the asking expertise (e_ask) and answering expertise (e_ans) of its answerer u_i:

$$\begin{aligned} qsocre_{\langle model \rangle}(q, a) &= \sigma \cdot e\_ask(u_i, q) \\ &+ (1 - \sigma) \cdot e\_ans(u_i, q), \end{aligned} \tag{2.27}$$

where $\sigma \in [0, 1]$ is a weighting parameter. With different models we can get different quality scores. In [138], four models are proposed to calculate an answer's quality score.

Wang et al. [146] proposed an analogical reasoning-based approach to rank answers based on their quality. This approach assumes that answers are connected to their questions with various

[4] Area Under the ROC Curve.

types of latent links. For instance, positive links indicate high-quality answers, while negative links represent low-quality ones. Therefore, the quality of new answers can be estimated from conducting analogical reasoning between new question-answer linkages and previous relevant positive linkages. Through analogy, high quality answers will obtain higher similarities, while low quality ones will get lower scores.

This approach consists of two stages: an offline stage and an online stage. In the offline stage, a Bayesian logistic regression model is learned with both positive and negative QA pairs. This model estimates how likely a QA pair contains a good answer. In the online stage, a supporting set of positive QA pairs is retrieved from the CQA portals using the new question as a query, and this supporting set will be used to rank new QA pairs.

Ranking-based approaches obtain two advantages. First, they avoid feature extraction. Second, they naturally combine questions and corresponding answers as a whole. However, rankings of answers cannot directly distinguish high-quality answers from low-quality ones. Manual checking is required to determine the cut-off point of a ranking list.

2.7 Answer Summarization

The prominent characteristic of CQA service is that it allows every user to answer every question. Usually a question will receive multiple answers. For each question, although one answer will be selected or voted as the best answer, it does not mean that other answers are bad or useless for future questions. First, the best answer may be an "incomplete answer" [24], which means the best answer only solves a partial information need. Second, since "everyone knows something" [1], different answers may contribute to different aspects of information, and using a single best answer will miss other user-generated knowledge. For example, a question ask-

ing for hotel recommendations may obtain many replies, and the best answer may only cover a few of good hotels. According to the report of [89], no more than 48% of CQA best answers[5] are indeed the unique best answers.

Answer summarization aims to summarizes all received answers to a complete, structured, succinct, and quality answer. On the one hand, it provides the asker a comprehensive reply and saves the time to look through all answers. On the other hand, the summarized answer is more reusable than a single best answer for future relevant questions. Approaches for answer summarization in CQA services fall into three categories:

1. Constraint-based: Answer summarization is converted to an optimization problem with some constraints, such as coverage and quality.

2. Question type-based: Different summarization techniques are applied based on different types of questions.

3. Graph-based: Sentences of answers are constructed to a graph, on which some classification algorithms are employed to determine whether to select a sentence to the summarized answer.

2.7.1 Constraint-based Summarization

Tomasoni and Huang [141] casted the answer summarization problem to a query-biased multi-document summarization task, and proposed a metadata-aware algorithm. The idea is to maximize the concept score in final summarization, in which concepts are represented as bag-of-BEs[6]. Four factors are applied on the concept scoring: quality, coverage, relevance, and novelty. Let c denote a concept, $S(c_i)$ be the concept score of concept c_i (which is estimated from

[5]This conclusion is derived from 400 randomly selected questions in four top categories of Yahoo! Answers.

[6]A BE is "a head|modifier|relation triple representation of a document developed at ISI" [164].

quality, coverage, relevance, and novelty), s_j represent a sentence, $length(j)$ denote the length of s_j, and M be the set of selected sentences for answer summarization, the summarization is then converted to a linear programming problem with constraints:

$$\begin{aligned}
\text{maximize} \quad & \sum_i S(c_i) \cdot x_i \\
\text{subject to} \quad & \sum_j length(j) \cdot s_j \leq L; \\
& \sum_j y_j \cdot occ_{ij} \geq x_i, \ \forall i; \\
& occ_{ij}, x_i, y_i \in \{0, 1\}, \ \forall i, j; \\
& occ_{ij} = 1 \text{ if } c_i \in s_j, \ \forall i, j; \\
& x_i = 1 \text{ if } c_i \in M, \ \forall i; \\
& y_j = 1 \text{ if } s_j \in M, \ \forall j.
\end{aligned}$$

The integer indicators x_i and y_j represent whether c_i and s_j are included in M. L is the length constraint which is a function of the lengths of all answers weighted by a corresponding quality score. Through solving the program, selected sentences compose the final summarization.

Liu et al. [85] took a similar approach with different concept scoring approaches. In [85], $S(c_i)$ is obtained by two random walk models. In the first content-based model,

$$\begin{aligned}
S(c_i) \ &= \ (1-d) \cdot \sum_{c_k \in C} \frac{sim(c_i, c_k)}{\sum_{c_j \in C} sim(c_i, c_j)} \cdot p(c_k|q) \\
&+ \ d \cdot \frac{rel(c_i|q)}{\sum_{c_j \in C} rel(c_j|q)}
\end{aligned} \tag{2.28}$$

where $d \in [0, 1]$ is a damping factor, C denotes all concepts, $p(c_k|q)$ means the proportion of c_k in q, $rel(c_i|q)$ represents the maximum similarity between c_i and all concepts in q, and $sim(c_i, c_k)$ denotes

the similarity between two concepts c_i and c_k [101]. In the second model, user authority is incorporated:

$$\begin{aligned} S(c_i) &= (1-d) \cdot \sum_{c_k \in C} \frac{sim(c_i, c_k|u_i, u_k)}{\sum_{c_j \in C} sim(c_i, c_j|u_i, u_j)} \cdot p(c_k|q) \\ &+ d \cdot \frac{rel(c_i|q)}{\sum_{c_j \in C} rel(c_j|q)}, \end{aligned} \tag{2.29}$$

where

$$\begin{aligned} sim(c_i, c_j|u_i, u_j) &= sim(c_i, c_j|u_i) + sim(c_i, c_j|u_j) \\ &= sim(c_i, c_j) \cdot auth(u_i) \\ &+ sim(c_i, c_j) \cdot auth(u_j). \end{aligned} \tag{2.30}$$

In the above equation, $auth(u_i)$ measures the authority score of u_i. The Hyperlink-Induced Topic Search (HITS) algorithm [64] is employed to calculate authority scores from a user-user graph based on QA activities.

In addition, an integer programming problem is constructed to select sentences for summarized answers. However, similar concepts may be chosen if they have high weights. To solve this problem, they further proposed a concept group-based objective function to eliminate redundant concepts and sentences.

2.7.2 Question Type-based Summarization

There is little literature on question-type based answer summarization. Liu et al. [89] focused on summarizing *open* and *opinion* questions as "they occupy more than half of the CQA questions". For *open* questions, a common multi-document summarization algorithm [52, 76] is employed, using a process of answer clustering, topic identification, fusion, and summary generation. For *opinion* questions, two subcategories are defined: *sentiment-oriented* and *list-oriented*:

1. For *sentiment-oriented* questions, answer sentences are first classified into *Support*, *Against*, or *Neutral* categories. The summarization is composed of two parts. In the first part, the sentences under each category are listed. In the second part, the count of sentences at each category is presented to show the ratio of different opinions.

2. For *list-oriented* questions, the summarization simply counts the number of different answers, and presents each answer together with its frequency.

2.7.3 Graph-based Summarization

Chan et al. [24] proposed a general Conditional Random Field (CRF) based method with group L_1 regularization for answer summarization. This method converts the answer summarization problem to a sequential labeling problem, which can be solved by the probabilistic graphical model. Let $\mathbf{x}$ denote the sentence sequence from all answers to a question, $\mathbf{y}$ be the corresponding labels, if $y_i = 1$, x_i will be incorporated in the summarized answer. According to CRF, the conditional probability of $\mathbf{y}$ given $\mathbf{x}$ is:

$$\begin{aligned} p(\mathbf{y}|\mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \exp\Big(\sum_{v \in V, l} \mu_l g_l(v, \mathbf{y}|_v, \mathbf{x}) \\ &+ \sum_{e \in E, k} \lambda_k f_k(e, \mathbf{y}|_e, \mathbf{x})\Big), \end{aligned} \tag{2.31}$$

where V and E represent the node set and the edge set, respectively. In addition, $\frac{1}{Z(\mathbf{x})}$ is the normalization factor, g_l is the feature function of the node l (sentence l), f_k is the function of edge k (modeling the interactions between sentences), μ and λ are weights, and $\mathbf{y}|_v$ ($\mathbf{y}|_e$) denotes the component of $\mathbf{y}$ related to node v (edge e). In [24], seven text features and six non-text features are utilized in modeling g_l, and four contextual factors are considered for modeling f_k. With

the assumption that some features are more important than others, group L_1 regularization is introduced to select salient features.

Given training data $D = \{(x^1, y^1), ..., (x^N, y^N)\}$, the parameters $\theta = (\mu_l, \lambda_k)$ of the CRF model are estimated from the following optimization problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{N} \log(p_\theta(y^i|x^i)) - C\sum_{g=1}^{G} \alpha_g \\ \text{subject to} \quad & \alpha_g \geq \| \vec{\theta_g} \|_2, \forall g. \end{aligned}$$

In the above optimization problem, G denotes the number of feature groups, $\vec{\theta_g}$ represents the parameters corresponding to the particular group g, and C controls the penalty magnitude of the parameters. During the inference, the labeling sequence is obtained by maximizing the probability of y given x:

$$y^* = \arg\max_y p_\theta(y|x). \tag{2.32}$$

Pande et al. [113] extended the above work, and employed a structured determinant point process (SDPP) to generate answer summarization. Using the similar graph in [24], SDPP finds "a path containing nodes that are individually coherent and together cover the most important information from the graph" [113].

2.7.4 Summary

Table 2.4 compares the above three approaches in terms of completeness, concision, summation, readability, and quality. Constraint-based summarization usually provides a complete and quality summarization. However, concision of summarized answer depends on constraint settings. For instance, the approach in [141] may face a redundancy problem if two similar concepts both get high scores. Liu et al. [85] alleviate the problem through adding a concept group constraint. Constraint-based summarization selects sentences as the

Table 2.4: Comparison among different answer summarization approaches.

Approach	Completeness	Concision	Summation	Readability	Quality
Constraint-based	Good	Constrain dependent	Sentence level	Not considered	Good
Question type-based	Good	Type dependent	Answer and sentence level	Good	Not considered
Graph-based	Good	Not considered	Sentence level	Not considered	Good

final answer, rather than summarize them. As such, the sequences of sentences are accordant to their original answers, which may be unstructured and incoherent, and reduces the readability of summarized answer.

Question type-based summarization is question-dependent, and takes different approaches for different questions. For instance, it takes answer-level summarization for open questions, while sentence-level summarization for sentiment-oriented questions. Therefore, the summarizations are usually well-structured and easy to read. However, this approach depends on question classification, and requires new summarization strategy when new question type appears. In addition, answer quality is not considered in summarization.

Graph-based summarization considers sentence importance and an inter-sentence relationship, which leads to good quality and complete summarization. The disadvantages of this approach are two-fold. First, it focuses on selecting sentences without considering sentence structure. Second, it may suffer from the redundancy problem since the relationships among selected sentences are not incorporated.

□ **End of chapter.**

Chapter 3

Question Popularity Analysis and Prediction

3.1 Motivation and Problem

The prosperity of CQA is accompanied by a huge amount of questions and answers. A series of studies is conducted to investigate answers in CQA so as to screen for better answers, such as quality analysis [62, 91], prediction [12, 47, 155], and ranking [7, 8]. But as for questions, fewer studies have so far been documented. In fact, questions in CQA vary in attracting answerer attention, answering attempts, and best answers. Taking questions from Yahoo! Answers as an example, some questions acquire thousands of tag-of-interests and answering attempts while some questions fail to get any answering attempts, indicating varied degrees of popularity.

The significance of investigating question popularity (QP) in CQA is three-fold:

1. *Low-popularity questions hinder CQA services*. Low-popularity questions, such as commercial advertisements, reduce user experience greatly.

2. *High-popularity questions promote the development of the community*. Since high-popularity questions induce more users to contribute their knowledge, they not only improve the effi-

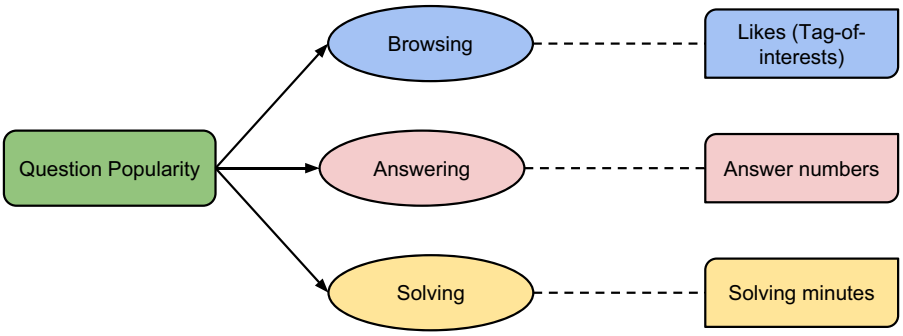

Figure 3.1: Construct of question popularity in CQA.

ciency of solving questions, but also enrich the knowledge base of the community.

3. *QP facilitates question retrieval and question recommendation*. We will improve question retrieval and question recommendation in CQA services if taking QP into account.

QP involves three dimensions (see Figure 3.1): (1) user attention; (2) answering attempt; and (3) best answer. In other words, high-popularity questions are supposed to attract significant user attention, induce more answer attempts, and receive best answers within a short period. Otherwise, questions failing to achieve the three criteria are labeled as low-popularity questions since the questions neither meet user needs nor contribute to the knowledge base of the community. It is worth noting that question utility and question difficulty are different from our QP here. The former aims to investigate the usefulness of questions [137], and the latter focuses on selecting the more difficult question from a question pair [79].

This chapter has five sections. In Section 3.2, we present the experimental data and the ground truth. Next, two studies are reported in Sections 3.3 and 3.4, respectively. The first study applies statistical analyses to find factors affecting QP. Based on the findings of Study 1, Study 2 proposes a novel graph-based semi-supervised learning (SSL) algorithm and applies the algorithm to predict QP. A

Table 3.1: Summary of questions and askers in *Entertainment* & *Music* category and its subcategories.

Subcategory	Number of questions	Number of askers
Celebrities	11,817	7,087
Comics & Animation	11,327	6,801
Horoscopes	7,235	2,203
Jokes & Riddles	3,685	2,569
Magazines	548	462
Movies	15,121	10,996
Music	32,948	18,589
Other - Entertainment	2,244	2,003
Polls & Surveys	138,507	18,685
Radio	640	272
Television	14,477	10,146
All	238,549	62,853

summary is given in Section 3.5.

3.2 Data Description

In this section, we first describe our data set. Then we detail how to set the ground truth for QP, providing the baseline for the following studies and analyses.

3.2.1 Data Set

We collect 238,549 resolved questions from July 7, 2010 to September 6, 2010 under the *Entertainment* & *Music* category of Yahoo! Answers. For each question, we extract both question information (question subject and content, post time, best answer post time, number of answers, and number of tag-of-interests by other users) and asker information (total points, number of answers, number of best

answers, number of questions asked, number of questions resolved, and number of stars received). There are altogether 11 subcategories under *Entertainment & Music*, and Table 3.1 gives the statistics of the data set.

3.2.2 Ground Truth Setting

We set the ground truth using the construct of QP in CQA (see Figure 3.1). To quantify the three variables, we employ the number of tag-of-interests (NT, reflecting the attractiveness of a question), the number of answers (NA), and the reciprocal of the minutes for getting the best answer (RM).

We first attempt to cluster these questions but the clustering results are not congruent with different seeds. In spite of this, the size of each cluster varies sharply from less than 10 to more than 50,000. Having consulted domain experts, we resort to expert-based reasoning. We calculate the Pearson correlation coefficient between each of the two variables, and find that NT and NA are correlated (0.500). But either NT or NA show little correlation with RM (-0.011 and 0.213, respectively). Therefore, we first normalize and average the values of NT and NA before converting them into an integer in a scale from 1 to 4 (NTA hereafter, with 4 being the highest popularity) using three equidistant cutting points of 0.75 (top 25%), 0.50 and 0.25 to assign each band with roughly the same amount of questions[1]. At the same time, RM is also transformed into 1 to 4 scale data using such approach. After that, two scale data are reasoned based on the rule base (see Table 3.2), which comes from consensus among the authors and domain experts. In the end, all questions are labeled as from level 1 to level 4, with level 4 featuring the highest popularity questions. Table 3.3 summarizes questions with levels, which are taken as the ground truth.

[1] As [152] and [120] suggested, we construct a balanced data set for predicting QP in the second study.

Table 3.2: Rule base for ground truth setting.

RM \ NTA	**4**	**3**	**2**	**1**
4	4	4	3	2
3	4	3	3	2
2	3	3	2	1
1	2	2	1	1

Table 3.3: Summary of questions in four levels.

Level	1	2	3	4
Count	53,806	62,192	69,836	52,715

3.3 Factors Affecting QP

In CQA portals, askers post questions on different topics. Therefore, askers and topics are probably the main sources of varied QP. However, we know little about the contribution of askers and topics to QP. Here, we are concerned with which factor has the most significant impact on QP, and we use the subcategories under *Entertainment* & *Music* as various topics. We do not select different categories as topics. First, we observe that the majority of users only ask questions in very few categories, thus choosing subcategories as topics are more representative. Second, different subcategories also reflect various topics, for instance, music and movies are two distinctive aspects of entertainment.

Study 1 is designed as follows. We first select the two most popular subcategories[2] (namely, *Music* and *Movies*, see Table 3.1) as two representative topics in Study 1 before checking their distributions of QP. Next, we track askers with at least five questions in both these

[2]The subcategory *Polls & Surveys* is not chosen since this subcategory is used to elicit public opinion. We observe questions in this subcategory usually receive much more answers than others.

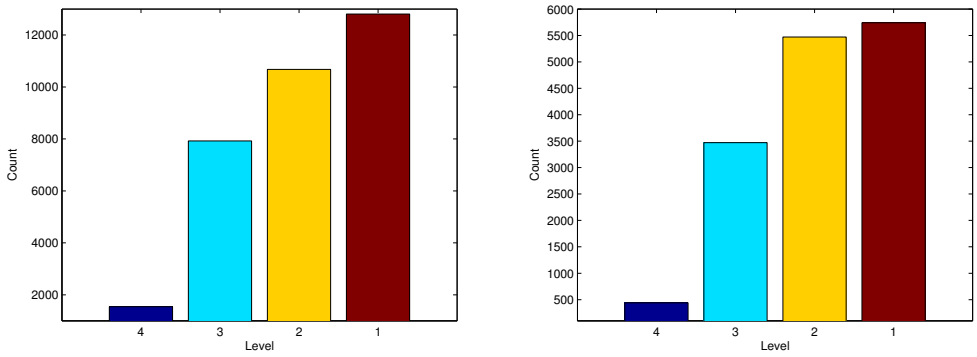

Figure 3.2: Distributions of QP in two topics. Left: *Music*; Right: *Movies*.

two subcategories and test QP of these questions.

Figure 3.2 presents the histograms of QP in *Music* and *Movies*. We can find that the distributions of QP in *Music* and *Movies* are close: the number of questions increases as QP decreases from level 4 to level 1, and the proportions of each level's questions remain similar. The difference lies in that the proportion of questions in level 2 of *Movies* is larger. This observation tells us topics alone cannot distinguish high-popularity questions from low-popularity ones.

To investigate the influence of askers, we select a total of 22 askers who have asked at least 5 questions in the two sub-categories. Mean and standard deviation (SD) of QP are reported in Table 3.4. Our observations are: (1) different askers own various QP on the same topic. For instance, the QP of user 8 is much higher than that of user 16; and (2) the QP of the same asker on various topics has significant differences. For instance, user 14 asks many high-popularity questions about *Movies*, but his QP in *Music* is poor. Therefore, we observe that it is the interaction between asker and topics which plays the most import role in distinguishing high-popularity questions from low-popularity ones.

To sum up, Study 1 examines the effects of askers and topics on QP. We observe that topics themselves cannot determine QP, and the interaction between askers and topics is the most important fac-

Table 3.4: Summary of QP for different askers.

User	Music		Movies		User	Music		Movies	
	Mean	SD	Mean	SD		Mean	SD	Mean	SD
1	2.50	0.93	2.17	0.41	12	2.48	0.95	2.47	0.84
2	2.45	0.52	2.57	0.98	13	2.84	0.68	2.83	0.41
3	1.86	0.90	1.45	0.82	14	1.33	0.52	2.40	0.89
4	2.65	0.72	2.60	0.55	15	1.90	0.74	1.83	0.75
5	1.90	0.74	2.00	0.71	16	1.80	0.84	1.83	0.75
6	2.62	0.87	1.83	0.86	17	2.15	0.55	2.50	1.05
7	2.48	0.68	2.20	0.84	18	2.36	0.92	1.67	0.87
8	2.86	0.92	2.14	0.90	19	2.00	1.00	2.00	1.00
9	2.38	0.92	2.30	1.06	20	2.00	0.67	2.00	1.00
10	2.50	0.53	2.40	0.55	21	2.69	0.68	2.80	0.45
11	2.00	0.71	1.50	0.55	22	2.13	0.99	2.57	1.27

tor affecting QP. This observation motivates us to design a novel algorithm to predict QP in the next study.

3.4 Prediction of QP

Study 1 has uncovered the main factors of QP, but it takes place when questions are resolved. In Study 2, we face an even more challenging prediction task: estimating QP right after a question is posted but not yet answered by any answerer. Motivated by the result of Study 1, we model the relationships among questions, topics, and askers as a bipartite graph model. Figure 3.3 shows one example, where u_1, u_2, and u_3 ask five questions $(q_1, ..., q_5)$ in three topics $(t_1, t_2,$ and $t_3)$. Each edge linking an asker and a question represents the question asked by the asker, and each rectangle denotes a topic. In the example, we know that u_1 asks q_1 and q_3, and q_2 is in topic t_1. Here topics are represented by subcategories or categories in CQA

portals.

The ideas of our algorithm are as follows:

1. As for the same topics, questions with similar structures and expressions will have similar QP[3], and askers with same profiles will embrace approximate expertise.

2. As for different topics, askers' ability to ask high-popularity questions is not equivalent and such ability is constant within a particular period.

3. Each question's QP is estimated from the quality of similar questions and the asker's ability to ask high-popularity questions in that topic. Meanwhile, each asker's ability of asking high-popularity questions at one topic is estimated from his QP and similar askers' ability in that topic.

Based on the these, we propose a graph-based SSL algorithm called "Mutual Reinforcement Label Propagation" (MRLP) to predict QP in CQA services. Before introducing MRLP, we first give the formal definitions of QP and asker expertise.

Definition 1 (**Question Popularity**). *Question q_i's popularity is represented by $\hat{q}_i$, which refers to its ability to attract user attention, get answering attempts, and receive the best answer efficiently. It ranges from 0 to 1. The higher the value is, the higher the popularity of the question.*

Definition 2 (**Asker Expertise**). *User u_j's expertise in topic t_k is represented by $\hat{u}_{jk}$, which reflects the user's ability to ask high-popularity questions within that topic. $\hat{u}_{jk}$ ranges from 0 to 1. It is worth noting that $\hat{u}_{jk}$ models the effect of interaction between the asker and the topic.*

[3]In fact, this assumption is quite weak since it does not consider any semantics.

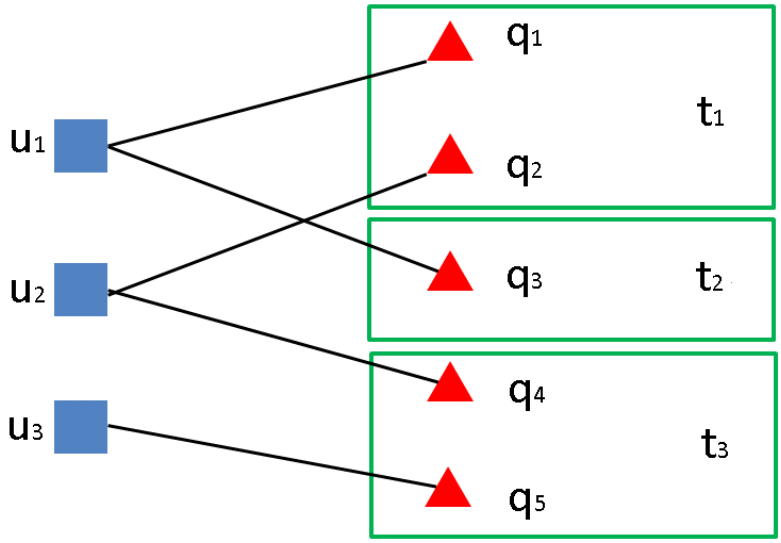

Figure 3.3: A toy example. Left: askers; Right: questions in three topics.

3.4.1 Algorithm

Suppose there are m askers who ask n questions in t topics. Let $U^1, U^2, ..., U^t$ denote the vectors of asker expertise in these topics, and Q denote the vector of QP. We define an $m \times n$ matrix E, where $e_{ij} = 1(i \in [1, m], j \in [1, n])$ means u_i asks q_j, otherwise $e_{ij} = 0$. From E we get E':

$$E'_{ij} = \frac{e_{ij}}{\sum_{k=1}^{n} e_{ik}}. \tag{3.1}$$

For the question part of the bipartite graph, we create edges between any two questions within same topics. The weight for the edge linking q_i and q_j is represented by $w(q_i, q_j)$, which is calculated using the cosine similarity between the features of two questions $\mathbf{x}_i$ and $\mathbf{x}_j$:

$$w(q_i, q_j) = \exp(-\frac{||\mathbf{x}_i - \mathbf{x}_j||^2}{\lambda_q^2}), \tag{3.2}$$

where λ_q is a weighting parameter. $w(q_i, q_j)$ is set to be 0 if q_i and q_j belong to two different topics. In addition, we define $w(q_i, q_i) = 0$.

Algorithm 1 MRLP

Input: asker expertise vector U_0^k, QP vector Q_0^k, E, transition matrixes M and N, weighting coefficients α and β, some manual labels of U_0^k and/or Q_0^k.
Output: U^k and Q^k.

1: Set $c = 0$.
2: **while** not convergence **do**
3: Propagate user expertise. $U_{c+1}^k = \alpha \cdot M \cdot U_c^k + (1-\alpha) \cdot E' \cdot Q_c^k$.
4: Propagate QP. $Q_{c+1}^k = \beta \cdot N \cdot Q_c^k + (1-\beta) \cdot E^T \cdot U_{c+1}^k$, where E^T is the transpose of E.
5: Clamp the labeled data of U_{c+1}^k and Q_{c+1}^k.
6: Set $c = c + 1$.
7: **end while**

Then, we define an $n \times n$ probabilistic transition matrix N:

$$N_{ij} = P(q_i \to q_j) = \frac{w(q_i, q_j)}{\sum_{k=1}^{n} w(q_i, q_k)}, \tag{3.3}$$

where N_{ij} is the probability of transition from q_i to q_j. Similarly, we create edges between any two askers who have asked questions in the same topic(s) for the asker part of the graph with λ_a as the weighting parameter using Eq. (3.2). Furthermore, we define an $m \times m$ probabilistic transition matrix M like N in Eq. (3.3).

For topic t_k, given some known labels of U_k and/or Q, we describe the MRLP in Algorithm 1. The equation at line 3 estimates each asker's expertise from his neighbors' expertise and his questions' popularity. Correspondingly, the equation at line 4 calculates each question's popularity from their neighbors' popularity and the corresponding asker's expertise. Repeating MRLP k times, question popularity and asker expertise are estimated.

Now, we prove the convergence of the algorithm.

Proof. Suppose there are l labeled data and u unlabeled data of question quality together with x labeled data and y unlabeled data of asker expertise, i.e., $Q^k = [\hat{q}_1^k, ..., \hat{q}_l^k, \hat{q}_{(l+1)}^k, ..., \hat{q}_{(l+u)}^k]^T$ and $U^k = [u_1^k, ..., u_x^k, u_{(x+1)}^k, ..., u_{(x+y)}^k]^T$. Thus, We can split E', E^T, M and N

into four parts:

$$E' = \begin{bmatrix} E'_{xl} & E'_{xu} \\ E'_{yl} & E'_{yu} \end{bmatrix},\ E^T = \begin{bmatrix} E^T_{xl} & E^T_{yl} \\ E^T_{xu} & E^T_{yu} \end{bmatrix},$$

$$M = \begin{bmatrix} M_{xx} & M_{xy} \\ M_{yx} & M_{yy} \end{bmatrix},\ N = \begin{bmatrix} N_{ll} & N_{lu} \\ N_{ul} & N_{uu} \end{bmatrix}.$$

Thus, we get

$$\begin{bmatrix} U^k_x \\ U^k_y \end{bmatrix}_{c+1} = \alpha \begin{bmatrix} M_{xx} & M_{xy} \\ M_{yx} & M_{yy} \end{bmatrix} \begin{bmatrix} U^k_x \\ U^k_y \end{bmatrix}_c + (1-\alpha) \begin{bmatrix} E'_{xl} & E'_{xu} \\ E'_{yl} & E'_{yu} \end{bmatrix} \begin{bmatrix} Q^k_l \\ Q^k_u \end{bmatrix}_c,$$

and

$$\begin{bmatrix} Q^k_l \\ Q^k_u \end{bmatrix}_{c+1} = \beta \begin{bmatrix} N_{ll} & N_{lu} \\ N_{ul} & N_{uu} \end{bmatrix} \begin{bmatrix} Q^k_l \\ Q^k_u \end{bmatrix}_c + (1-\beta) \begin{bmatrix} E^T_{xl} & E^T_{yl} \\ E^T_{xu} & E^T_{yu} \end{bmatrix} \begin{bmatrix} U^k_x \\ U^k_y \end{bmatrix}_c.$$

Since U^k_x and Q^k_l are clamped to manual labels in each iteration, we now only consider U^k_y and Q^k_u. From the above two equations we get:

$$\begin{aligned} \begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix}_{c+1} &= \begin{bmatrix} \alpha M_{yy} & (1-\alpha)E'_{yu} \\ (1-\beta)E^T_{yu} & \beta N_{uu} \end{bmatrix} \begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix}_c \\ &+ \begin{bmatrix} \alpha M_{yx} U^k_x + (1-\alpha)E'_{yl} Q^k_l \\ \beta N_{ul} Q^k_l + (1-\beta)E^T_{xu} U^k_x \end{bmatrix}. \end{aligned}$$

Let

$$A = \begin{bmatrix} \alpha M_{yy} & (1-\alpha)E'_{yu} \\ (1-\beta)E^T_{yu} & \beta N_{uu} \end{bmatrix},\ b = \begin{bmatrix} \alpha M_{yx} U^k_x + (1-\alpha)E'_{yl} Q^k_l \\ \beta N_{ul} Q^k_l + (1-\beta)E^T_{xu} U^k_x \end{bmatrix},$$

we get

$$\begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix}_n = A^n \begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix}_0 + (\sum_{i=1}^{n} A^{i-1})b,$$

where $\begin{bmatrix} U_y^k \\ Q_u^k \end{bmatrix}_0$ are the initial values for unlabeled askers and questions. The following proof is similar to the one in the second chapter of [168]. Since M, N, E' and E^T are row normalized (each row of E^T only contains one "1", others are "0"), M_{yy}, N_{uu}, E'_{yu}, and E^T_{yu} are sub-matrices of them,

$$\exists \gamma < 1, \sum_{j=1}^{y+u} A_{ij} \leq \gamma, \forall i = 1, ..., y+u.$$

So

$$\begin{aligned} \sum_j A_{ij}^n &= \sum_j \sum_k A_{ik}^{n-1} A_{kj}, \\ &= \sum_k A_{ik}^{n-1} \sum_j A_{kj}, \\ &\leq \sum_k A_{ik}^{n-1} \gamma, \\ &\leq \gamma^n, \end{aligned}$$

which means the sum of each row of A converges to zero. Therefore, $A^n \begin{bmatrix} U_y^k \\ Q_u^k \end{bmatrix}_0 \to 0$. Finally, we get

$$\begin{bmatrix} U_y^k \\ Q_u^k \end{bmatrix} = (I - A)^{-1} b,$$

which are fixed values. □

When m and n are large values, it is inefficient to compute $(I - A)^{-1}$ directly. As $\sum_j A_{ij}^n \leq \gamma^n$, we can apply Algorithm 1 to calculate U^k and Q^k iteratively.

Table 3.5: Summary of data in Study 2.

Item	Music	Movies
Number of questions	7,373	1,076
Number of high-popularity questions	3,670	331
Number of low-popularity questions	3,703	745
Number of askers	314	56

3.4.2 Experimental Setup

To verify the effectiveness of the MRLP in predicting QP, we experiment with the data described in Section 3.2. For each topic of *Music* and *Movies*, we choose questions of those askers who asked at least 10 questions in that topic. Since our goal is to distinguish high-popularity questions from low-popularity ones, we follow the common binary classification setting in previous works [129, 88, 3]. Thus, we take questions of level 3 and level 4 as high-popularity ones and the other questions as low-popularity ones. Table 3.5 summarizes the data. To get prediction performance at different training levels, we adjust the training rates from 10% to 90% in our experiments.

Features. Referring to the works in [3] and [4], we adopt the features listed in Table 3.6 to construct graphs and train classifiers. They are divided into question-related and asker-related features. Question-related features are extracted from question texts including subject and content, and asker-related features come from asker profiles. For features such as POS_entropy, we use the tool OpenNLP[4] to conduct tokenization, detect sentences, and annotate the part-of-speech tags. In addition, we utilize the Microsoft Office Word Primary Interop Reference[5] to detect typographical errors.

We also report the information gain of each feature in Table 3.6.

[4]http://opennlp.sourceforge.net/

[5]http://msdn.microsoft.com/library/bb406008(v=office.11).aspx/

Table 3.6: Summary of features extracted from questions and askers.

Name	Description	IG
Question-related features		
Sub_len	Number of words in question subject (title)	0.0115
Con_len	Number of words in question content	0.0029
Wh-type	Whether the question subject starts with Wh-word	0.0001
Sub_punc_den	Number of question subject's punctuation over length	0.0072
Sub_typo_den	Number of question subject's typos over length	0.0021
Sub_space_den	Number of question subject's spaces over length	0.0138
Con_punc_den	Number of question content's punctuation over length	0.0096
Con_typo_den	Number of question content's typos over length	0.0006
Con_space_den	Number of question content's spaces over length	0.0113
Avg_word	Number of words per sentence in question's subject and content	0.0048
Cap_error	The fraction of sentences which are started with a small letter	0.0064
POS_entropy	The entropy of the part-of-speech tags of the question	0.0004
NF_ratio	The fraction of words that are not the top-10 frequent words	0.0009
Asker-related features		
Total_points	Total points the asker earns	0.0339
Total_answers	Number of answers the asker provided	0.0436
Best_answers	Number of best answers the asker provided	0.0331
Total_questions	Number of questions the asker provided	0.0339
Resolved_questions	Number of resolved questions asked by the asker	0.0357
Star_received	Number of stars received for all questions	0.0367

It is found that all features' information gains (IGs) are small, which means these features are not so salient to QP. In addition, asker-related features are more crucial than question-related features since their information gains are higher. As for question-related features, space density and subject length are the most important ones.

Methods for comparison. We compare our MRLP algorithm with the following methods:

- *Logistic Regression*. Shah et al. [129] apply logistic regression model to predict answer quality in Yahoo! Answers. Here we adopt the same approach to predict QP with question-related features alone (LR_Q), and both question-related and asker-

related features (LR_QA). These two methods are treated as baselines.

- *Stochastic Gradient Boosted Tree*. Agichtein et al. [3] report the stochastic gradient boosted trees [38] (SGBT) perform best among several classification algorithms, including SVM and log-linear classifiers, to classify content quality in CQA service. For SGBT classifier, in each iteration a new decision tree is built to fit a model for the residuals left by the classifier on the previous iteration. In addition, a stochastic element is added to each iteration to smooth the results and prevent overfitting. For different features we have SGBT_Q and SGBT_QA.

- *Harmonic Function*. Zhu et al. [169] propose the harmonic function algorithm for label propagation on a homogeneous graph, where all nodes represent the same kind of object. To estimate QP, we create a graph in which each node stands for a question and each edge's weight represents two questions' similarity. Let W denote the weight matrix and D denote the diagonal matrix with $d_i = \sum_j w_{ij}$, then we can construct the stochastic matrix $P = D^{-1}W$. Let $f = \begin{bmatrix} f_l \\ f_u \end{bmatrix}$ where f_l are the qualities of labeled questions and f_u are what we want to predict. We split the matrix W (also D and P) into four parts:

$$W = \begin{bmatrix} W_{ll} & W_{lu} \\ W_{ul} & W_{uu} \end{bmatrix}, \tag{3.4}$$

where W_{ll} means the similarities among labeled questions, and W_{lu} means the similarities between labeled questions and unlabeled questions. Similarly, W_{ul} represents the similarities between unlabeled questions and labeled questions, and W_{uu} represents the similarities among unlabeled questions. The harmonic solution is:

$$f_u = (D_{uu} - W_{uu})^{-1} W_{ul} f_l = (I - P_{uu})^{-1} P_{ul} f_l. \tag{3.5}$$

Similarly, we construct HF_Q and HF_QA using different features.

In our experiments we use the tool Weka [45] to build logistic regression models and SGBT classifiers, with default settings adopted. For graph-based algorithms such as HF_Q, HF_QA, and MRLP, we build 10-NN graphs. In addition, for each training rate, we randomly sample training data and testing data 10 times, and take the average as final results.

Evaluation metrics. We adopt accuracy, sensitivity, and specificity as the evaluation metrics. Accuracy reflects the overall performance of prediction, while sensitivity and specificity measure the algorithm's ability to classify high-popularity and low-popularity questions into their correct classes, respectively.

3.4.3 Experimental Results

Table 3.7 reports the accuracy of these methods under various training rates across two topics. Figure 3.4 and Figure 3.5 present sensitivity and specificity of each method in *Music*. Similar results are obtained in *Movies*.

With the increase number of training data, all methods perform better. Among these methods, MRLP performs much better than baseline methods (LR_Q and LR_QA) in all settings. For instance, when the training rate is 10% for *Movies*, the accuracy of MRLP is 20.04% and 19.54% higher than that of LR_Q and LR_QA, respectively. In addition, MRLP is more accurate in predicting QP than other compared methods. These results demonstrate that MRLP is more effective in predicting questions' qualities through modeling the interaction between askers and topics and capturing the mutual reinforcement relationship between asker expertise and QP.

Meanwhile, neither the MRLP nor other methods perform very well in classifying QP across the two topics. Even the training rate is set to be 90%, there are still more than 35% of questions not cor-

Table 3.7: Different methods' performance with question-related features alone versus both question-related and user-related features (*Music*: $\alpha = 0.2$, $\beta = 0.2$; *Movies*: $\alpha = 0.8$, $\beta = 0.1$).

Method	**Accuracy under training rate (%)**									
	Music					**Movies**				
	10	**30**	**50**	**70**	**90**	**10**	**30**	**50**	**70**	**90**
LR_Q	0.550	0.558	0.560	0.568	0.583	0.479	0.512	0.516	0.518	0.519
LR_QA	0.567	0.584	0.595	0.603	0.612	0.481	0.513	0.517	0.521	0.524
HF_Q	0.535	0.545	0.567	0.579	0.596	0.415	0.472	0.493	0.506	0.514
HF_QA	0.547	0.576	0.584	0.602	0.621	0.491	0.536	0.549	0.558	0.564
SGBT_Q	0.560	0.576	0.581	0.584	0.596	0.538	0.541	0.551	0.569	0.582
SGBT_QA	0.576	0.610	0.627	0.631	0.636	0.547	0.562	0.566	0.587	0.605
MRLP	**0.608**	**0.625**	**0.633**	**0.656**	**0.664**	**0.575**	**0.589**	**0.604**	**0.615**	**0.623**

rectly classified. The reason is that question text and asker profile features are not salient features of QP, as shown in Table 3.6. Since all features' information gains are less than 0.05, it is very hard to make satisfactory predictions using these features. This limitation leads us to further explore the salient features of QP in our future work.

Question-related features versus asker-related features. Comparing LR_QA, HF_QA, and SGBT_QA with LR_Q, HF_Q, and SGBT_Q from Table 3.7, we find that with asker-related features the accuracy of prediction is substantially higher than the same methods without using asker-related features in both *Music* and *Movies*. This observation shows that incorporating asker-related features increases the prediction performance. As explained in the first study, the interaction between topics and askers is the most deterministic factor for QP. Our feature analysis results are accordant to our observations.

Figures 3.4 and 3.5 report different algorithms' sensitivity and specificity at various training rates. Similarly, with the help of asker-related features, the same algorithms usually achieve better predic-

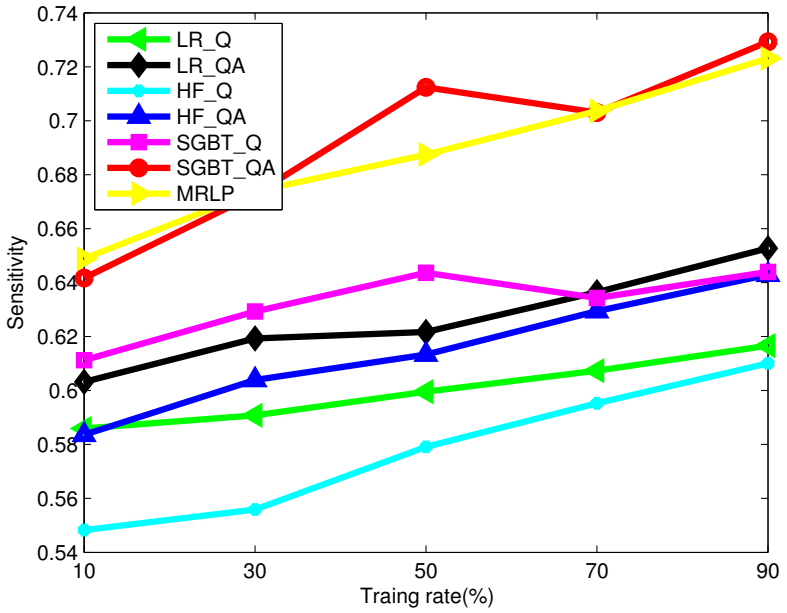

Figure 3.4: Sensitivity versus training rate across various methods in *Music*.

tions. For instance, when 50% of data are used as training data in *Music*, SGBT_QA's sensitivity is 10.67% higher than that of SGBT_Q, and 2.43% higher on specificity. Other algorithms obtain similar results at each training rate.

Mixture versus separation of user-related features. Comparing LR_QA, HF_QA, and SGBT_QA with MRLP, we observe that MRLP performs the best on accuracy. When looking at the sensitivity in Figure 3.4 and the specificity in Figure 3.5, MRLP is more balanced in sensitivity and specificity than other algorithms. For instance, SGBT_QA obtains the highest sensitivity for *Music* in most cases but relatively low specificity, meaning it predicts most questions as high-popularity ones. HF_QA obtains the high specificity for *Music* but very low sensitivity, and it almost predicts most questions as low-popularity ones. MRLP gives the most balanced performance since it integrates the question-related features with asker-related features naturally other than a simple combination. In partic-

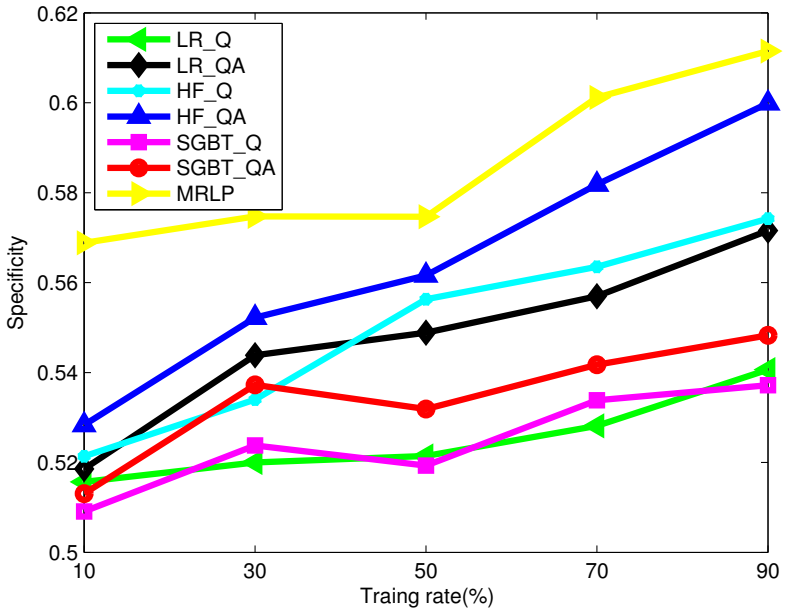

Figure 3.5: Specificity versus training rate across various methods in *Music*.

ular, it improves the accuracy of the second best method (SGBT_QA) by 3.5% in *Music* and 4.9% in *Movies*. MRLP naturally separates question-related features and user-related features in graph construction, and the above results demonstrate that this approach is better than simply combining these features. Therefore, MRLP is more effective in discriminating high-popularity questions from low-popularity ones.

3.5 Summary

In this chapter, we conduct two studies to investigate QP in CQA services. In Study 1, we analyze the factors influencing QP and find that the interaction of users and topics leads to the differences in QP. Based on the findings of Study 1, Study 2 proposes a mutual reinforcement-based label propagation algorithm to predict QP using the features of question texts and asker profiles.

We experiment with real world data sets and the results demonstrate that our algorithm is more effective in distinguishing high-popularity questions from low-popularity ones than the logistic regression model and other state-of-the-art algorithms. However, as current features extracted from question texts and asker profiles are not so salient, neither our algorithm nor other competitors achieves satisfactory performance at present.

Current results lead us to further explore the salient features of QP in the future work. In addition, semantics can be incorporated to improve the proposed algorithm.

□ **End of chapter.**

Chapter 4

Question Routing

4.1 Problem and Motivation

Since CQA portals are so popular, one interesting and important question is whether this service can solve questions effectively. To investigate this problem, we randomly sample 3,000 questions in both Yahoo! Answers and Baidu Zhidao to observe these questions' statuses. Table 4.1 reports the result, from which we can find that in Yahoo! Answers only 17.6% of questions receive satisfying answers within 48 hours. For those unresolved questions, nearly 1/5 of them receive no response (211 questions' statuses are missing since they are deleted by the system). For Baidu Zhidao, 22.7% of questions are well-resolved, which is higher than that of Yahoo! Answers. However, 42.8% of unresolved questions receive no response at all, and the rate is much higher than that of Yahoo! Answers. From the above observations we find that the efficiency of above two popular CQA portals is not good enough since a great number of questions are not resolved promptly.

To promote the efficiency of question answering, in this chapter we propose a framework of *Question Routing* (QR). This framework routes a newly posted question to answerers who are most likely to give answers in a short period. The concept of QR contains two meanings. First, it aims to find appropriate answerers who can provide high quality answers, i.e., these answerers must have

Table 4.1: Statuses of tracked questions two days after being posted.

CQA portal	Number of resolved questions	Number of unresolved but answered questions	Number of unanswered questions
Yahoo! Answers	527	1,820	442
Baidu Zhidao	682	1,325	993

expertise on the routed question. Second, answerers who receive the routed question must be able to provide quick responses, i.e., they are available to answer the question in time. Under the framework, we further propose several models to estimate answerer expertise and answerer availability.

This chapter proceeds as follows. Section 4.2 details the framework of QR and approaches to estimate answerer expertise and answerer availability. In Section 4.3 and Section 4.4, we describe the experimental setup and the analyses of results. Section 4.5 gives a summary of this chapter.

4.2 Question Routing Framework

The whole process of QR is illustrated in Figure 4.1. The operation includes two stages: answerer profiling (off-line) and routing (on-line). The first stage relates to building up answerer profiles in terms of the answering history of each answerer. At the stage of routing, expertise estimation detects the expertise of all answerers to new questions (answerer expertise) based on their answerer profiles. Answerer expertise is defined as follows:

Definition 3 (**Answerer Expertise**). *Answerer u's expertise on q, which is represented by $E(u, q)$, models the probability of the answerer u gives a good answer to question q. It is a score between*

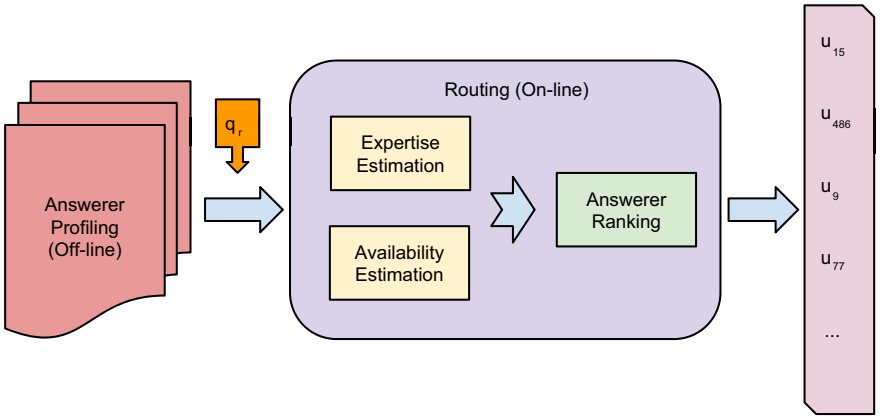

Figure 4.1: The framework of *Question Routing*.

0 and 1. A good answer should feature "responsiveness, accuracy, and comprehensiveness" (see [9]).

Meanwhile, availability estimation calculates each candidate's availability to answer questions within time T from the answerer's answering performance profiles. Answerer availability is defined as follows:

Definition 4 (**Answerer Availability**). *Answerer u's availability in a time period T, which is represented by $A(u, T)$, models the probability of the answerer u is available to provide answers for any questions during T. For convenience we simply assume u is available in T if he answers at least one question in this period.*

Finally, answerer ranking utilizes answerer expertise to rank the answerers, with top-*k* answerers being chosen as eligible answerers to new questions.

4.2.1 Answerer Profiling

In the phase of answerer profiling, we establish each answerer's performance profile from his answering history. For the answerer who has answered at least one question in CQA services, we use all questions he once answered and the corresponding answers provided by

the answerer to build his performance profile. Next, the performance profile will be used to estimate the expertise and availability on the new question q_r.

4.2.2 Expertise Estimation

In this section, we will present three approaches to estimate each answerer's expertise on q_r. The first one is the query likelihood language model (QLLM) which estimates how likely q_r can be generated from the questions each answerer has answered. The latter two models utilize the quality of answers to refine QLLM. In the following, we use $E(u_i, q_r)$ to denote answerer u_i's expertise on new question q_r, ranging from 0 to 1. The higher value $E(u_i, q_r)$ is, the higher expertise u_i has for q_r.

Expertise estimation without answer quality. Our QR concept is similar to Liu et al.'s work on identifying "the group of 'experts' who are likely to provide answers to given questions" [84]. In [84], several state-of-the-art language models are applied to ranking answerers' expertise for new questions. We adopt QLLM as our first model, which performs well in their experiments.

Here we briefly describe how to apply QLLM to rank potential answerers for given questions. Let q_{u_i} denote all previously answered questions by answerer u_i, for a new question q_r, u_i's expertise on q_r is defined as how likely q_r can be generated from q_{u_i}:

$$E(u_i, q_r) = P(q_r|q_{u_i}), \tag{4.1}$$

$$P(q_r|q_{u_i}) = \prod_{\omega \in q_r} P(\omega|q_{u_i}). \tag{4.2}$$

With Jelinek-Mercer smoothing [158],

$$P(\omega|q_{u_i}) = (1-\lambda)P_{ml}(\omega|q_{u_i}) + \lambda P_{ml}(\omega|C), \tag{4.3}$$

where C is the collection of all questions, and λ is a weighting coefficient to adjust the weight of smoothing. We set $\lambda = 0.8$ in

our experiments according to the empirical value presented in [158]. $P_{ml}(\omega|q_{u_i})$ is the maximum likelihood estimate of generating term ω from u_i's previously answered questions q_{u_i}:

$$P_{ml}(\omega|q_{u_i}) = \frac{tf(\omega, q_{u_i})}{\sum_{\omega' \in q_{u_i}} tf(\omega', q_{u_i})}, \tag{4.4}$$

where $tf(\omega, q_{u_i})$ means the term frequency of the term ω in q_{u_i}. Similarly,

$$P_{ml}(\omega|C) = \frac{tf(\omega, C)}{\sum_{\omega' \in C} tf(\omega', C)}, \tag{4.5}$$

where $tf(\omega, C)$ is the term frequency of the term ω in C.

Expertise estimation with answer quality. The above model assumes that the answerer has high expertise on a new question q_r if he has answered many similar questions before. However, it does not consider the quality of previous answers. An answerer may answers a great number of questions which are similar to q_r, but we cannot reach the conclusion that the answerer must be an expert for question q_r if most previous answers are of low quality. Even if we route q_r to this answerer, the asker may not get a satisfying answer. In order to obtain a more accurate prediction, we utilize answer quality in expertise estimation. Thus,

$$E(u_i, q_r) = \alpha \cdot P(q_r|q_{u_i}) + (1 - \alpha) \cdot Q(u_i, q_r), \tag{4.6}$$

where $Q(u_i, q_r)$ reflects u_i's answer quality for question q_r. In addition, $\alpha \in [0, 1]$ is a weighting coefficient.

We propose to utilize two models to estimate $Q(u_i, q_r)$ from previous answers' quality. The *Basic Model* is straightforward: it assumes an answerer's answer quality on the new question q_r is the weighted average answer quality of similar questions he answered

previously. It is defined as:

$$Q_{BM}(u_i, q_r) = \frac{\sum_{q_j \sim u_i} Q(u_i, q_j) \cdot sim(q_j, q_r)}{\sum_{q_j \sim u_i} sim(q_j, q_r)}, \tag{4.7}$$

where $q_j \sim u_i$ denotes the questions u_i has answered, and $sim(q_j, q_r)$ means the cosine similarity between question q_j and q_r.

We use the vector space model [125] to represent each question, i.e., $q = (w_{q_1}, w_{q_2}, ..., w_{q_T})^T$, where T is the number of terms (unique words) in all questions. w_{qt} is the tf-idf weight for the term t and it is defined as:

$$w_{qt} = tf_{t,q} \times \log \frac{N}{df_t}, \tag{4.8}$$

in which $tf_{t,q}$ is the term frequency of term t in question q, N is the number of all resolved questions in the archive, and df_t is the number of questions containing the term t. Thus, w_{qt} is zero if term t does not appear in question q.

The cosine similarity between question a and question b is calculated as follows:

$$sim(a, b) = \frac{\sum_{i=1}^{T} w_{ai} \cdot w_{bi}}{\sqrt{\sum_{i=1}^{T} w_{ai}^2} \sqrt{\sum_{i=1}^{T} w_{bi}^2}}. \tag{4.9}$$

However, this model may suffer from the data sparsity problem. Table 4.2 shows a toy example. Each row in Table 4.2 represents an answerer and each column represents a question. Each value in cell (i, j) is u_i's answer quality on question q_j, which is already calculated. Since most answerers only answered one question, the answerer-question matrix is very sparse. For instance, now we use Eq. (4.7) to estimate u_1's answer quality for the new question q_r. Since u_1 only answered q_2 before,

$$Q_{BM}(u_1, q_r) = \frac{Q(u_1, q_2) \cdot sim(q_2, q_r)}{sim(q_2, q_r)} = Q(u_1, q_2). \tag{4.10}$$

Table 4.2: Estimating answerers' answer quality using the *Basic Model*. Here only u_1's answer quality on q_2 (number in blue cell) is used to estimate u_1's answer quality on q_r.

Answerer \ Question	q_1	q_2	q_3	q_4	q_r
u_1		0.7			?
u_2		0.5			
u_3	0.9			0.8	
u_4			0.6		

So in this case u_1's answer quality on q_2 is used to estimate u_1's answer quality on q_r, no matter how similar it is between q_r and q_2.

In order to better utilize the known information, we borrow the idea of similarity fusion in collaborative filtering [143], which leverages other similar answerers' answer quality on similar questions to smooth the *Basic Model*. Let U denote the answerer set and $sim(u_i, u_j)$ be the cosine similarity between u_i and u_j, then in the *Smoothed Model*,

$$Q_{SM}(u_i, q_r) = \beta \frac{\sum_{q_j \sim u_i} Q(u_i, q_j) \cdot sim(q_j, q_r)}{\sum_{q_j \sim u_i} sim(q_j, q_r)} + (1-\beta) \frac{\sum_{u_j \in U/u_i} \sum_{q_k \sim u_j} Q(u_j, q_k) \cdot sim(Q(u_j, q_k), Q(u_i, q_r))}{\sum_{u_j \in U/u_i} \sum_{q_k \sim u_j} sim(Q(u_j, q_k), Q(u_i, q_r))}, \quad (4.11)$$

and

$$sim(Q(u_j, q_k), Q(u_i, q_r)) = \frac{1}{\sqrt{\frac{1}{sim(u_i, u_j)^2} + \frac{1}{sim(q_k, q_r)^2}}}. \quad (4.12)$$

Table 4.3: Estimating answerers' answer quality using the *Smoothed Model*. Here, not only u_1's answer quality on q_2 (number in blue cell), but similar answerers' answer quality on similar questions (numbers in yellow cells) are also used to estimate u_1's answer quality on q_r.

Question / **Answerer**	q_1	q_2	q_3	q_4	q_r
u_1		0.7			?
u_2		0.5			
u_3	0.9			0.8	
u_4			0.6		

Compared with the *Basic Model*, the *Smoothed Model* utilizes the answer quality of similar answerers on similar questions as extra sources to smooth the former estimation. As shown in Table 4.3, this model uses more answerers' answer quality on similar questions in smoothing.

We choose the features listed as follows to calculate the cosine similarity between two answerers:

- Number of total points the user owns;
- Number of answers the user has provided;
- Number of best answers the user has provided;
- Number of questions the user has asked;
- Number of stars the user received.

The next problem is how to determine the answerer's answer quality in previously answered questions. Given an answer $a(u_i, q_j)$ which is posted by u_i for question q_j, we use $\mathbf{a}_{ij}$ to denote the feature vector of $a(u_i, q_j)$. By letting the probability of $a(u_i, q_j)$ being a good answer be $P(\mathbf{a}_{ij})$, we use logistic regression to model the

log-odds of this probability, which is shown as follows:

$$\log(\frac{P(\mathbf{a}_{ij})}{1-P(\mathbf{a}_{ij})}) = \sigma^T \mathbf{a}_{ij}, \tag{4.13}$$

where σ is the coefficient vector of the regression model.

The following features are extracted in model training:

- Answer length;
- Question-Answer length ratio;
- Number of answers for this question;
- Number of times the answer is rated up other users;
- Number of times the answer is rated down by other users;
- The answerer's total points;
- The answerer's best answer ratio.

Following the work in [57], we apply feature conversion on the non-monotonic features using Kernel Density Estimation (KDE) [55]. In addition, we scale all feature values into the range of [0,1] using min-max normalization.

4.2.3 Availability Estimation

The motivation of availability estimation comes from the experience in daily life. Sometimes you want to ask one of your friends for help but unfortunately he is too busy to answer at that moment. Although this friend is able to solve your problem well, he cannot provide you with any help.

We assume one answerer is available to provide answers for the routed questions when he log on Yahoo! Answers, however, it is impossible to get users' login information for us. Thus, our objective is changed to estimating whether the answerer logs on in several

days after the routed question is posted. We model this problem as a typical trend analysis problem in time-series data mining and use an autoregressive model to make the forecasting. Formally, we use $A(u_i, t)$ to denote the probability that u_i is available at time t to answer routed questions (usually t represents one specific day). In practice, we set $A(u_i, t) = 1$ when u_i posted at least one answer on the day t, otherwise $A(u_i, t) = 0$.

The autoregressive model is presented as follows:

$$A(u_i, t) = \lambda_1 A(u_i, t-1) + ... + \lambda_p A(u_i, t-p) + \varepsilon, \qquad (4.14)$$

where p is a time parameter, ε is the source of randomness, and λs are parameters of the model. The latent assumption of the autoregressive model is that the probability that u_i is available at day t is a linear combination of the availability in earlier p days. Given a group of training data $\{A(u_i, t), A(u_i, t-1), ..., A(u_i, t-p)\}$ $(i = 1, ..., m)$ where m is the number of answerers, we can estimate the value of $\lambda_1, \lambda_2, ..., \lambda_p$. Then we can apply the above model to predict the value of $A(u_i, t)$ when given $A(u_i, t-1), ..., A(u_i, t-p)$.

Therefore, each answerer's availability for a period of time $T = \{t_1, ..., t_s\}$ is calculated as:

$$A(u_i, T) = 1 - \prod_{j=1}^{s} (1 - A(u_i, t_j)). \qquad (4.15)$$

4.2.4 Answerer Ranking

We treat answerer expertise on q_r and answerer availability in a range of time T as independent, and use their linear combination as the final QR score for each answerer:

$$QR(u_i, q_r, T) = \gamma \cdot E(u_i, q_r) + (1 - \gamma) \cdot A(u_i, T), \qquad (4.16)$$

where $\gamma \in [0, 1]$ is a weighting parameter. We rank all answerers according to their QR scores and put top-k candidates into the final QR list.

To sum up, we present the whole procedure of QR in Algorithm 2.

Algorithm 2 Routing questions to appropriate answerers in CQA

Input: Resolved questions, answers, answerers, new question q_r, time period T, K, α, β, and γ.

Output: QR list L for q_r.

1: For each answerer in the archive, build his performance profile from his answering history.
2: Estimate each answerer's expertise on q_r using Eq. (4.1) or Eq. (4.6).
3: Calculate each answerer's availability within T after q_r is posted using Eq. (4.15).
4: Ranking all answerers according to their QR scores which are calculated from Eq. (4.16) and put top-k candidates into the QR list L.

4.3 Experimental Setup

We conduct a series of experiments using Yahoo! Answers data, and apply the proposed QR framework to resolved questions (testing data) to explore whether the answerers in the QR list actually answer those questions. We aim to investigate the answers to those research questions through experiments:

1. Is it reasonable to route new questions to those answerers who have answered similar questions? (i.e., do answerers prefer to answer the question that some similar questions they have answered?)

2. What's the influence of answer quality to the performance of QR?

3. Does the *Smoothed Model* give better answer quality estimation and QR performance improvement?

4. Is it useful to estimate answerer availability in QR or does availability estimation improve the performance by much?

In the following, we first present our data set. Next, we describe all compared approaches in our experiments. Then we describe the evaluation metrics which are used to evaluate the performance of each method in QR.

4.3.1 Data Collection

Our data set is a snapshot of resolved questions from April 6, 2010 to May 14, 2010 under the *Computers & Internet* category of Yahoo! Answers. For each question, we crawled information for questions (subjects), answers (content, number of rate-ups, number of rate-downs) and answerers (total points, number of answers, number of best answers, number of questions asked, and number of stars received). For each answerer, we also crawled his answer history log during this time if it is public. The stopwords in question subjects and answer content are removed. In our experiments, the questions posted after May 6, 2010 are treated as new questions to be routed (Set A, testing data) and the ones remaining are treated as archive data (Set B). The ground truth for each question in Set A are the answerers who actually answer it.

Data set. Table 4.4 presents the information of the data set. After splitting as stated above, Set A includes 2,564 questions, 7,059 answers, and 3,952 answerers. Set B includes 17,182 questions, 48,663 answers, and 16,298 answerers.

Automatic labeling. Recall that we use a logistic regression model to estimate each answerer's expertise on the questions which the answerer has answered (i.e., each previous answer's quality). We adopt the community's and the askers' choices to avoid manual labeling. Answers are labeled as "good" and "bad" as follows. For each question in Set B, the answer is labeled as a "good" answer (the probability for the answer being good is high) only the following two conditions are met:

1. It is selected as the best answer;

Table 4.4: Description of our data set.

Item	Value
number of questions	19,746
average question length in words	10.54
average number of answers for one question	2.82
maximum number of answers for one question	31
number of answers	55,722
average answer length in words	55.35
number of askers	16,036
number of answerers	19,280
number of both askers and answerers	1,780
number of answerers only	17,500
number of askers only	14,256

2. It obtains more than 50% of rate-ups for all answers of the question.

Meanwhile, one answer is labeled as a "bad" answer if it receives more than 50% of rate-downs for all answers of the question. As such, 2,153 "good" instances and 2,593 "bad" instances served as training data to estimate the parameters of the logistic regression model.

T and p in availability estimation. In order to determine the value of T in availability estimation, we analyze the duration from the question being posted to the last answer being provided for questions in Set B. It shows that the maximum duration time for a question to receive an answer is 2.16 days, the minimum value is 14 seconds, and the mean duration is 2.21 hours. Based on these observations, we set $T = 3$ in our experiments in order to receive all possible answers. Moreover, we set $p = 3$ which means the first three days' answering records are used to estimate the answerer's availability in the fourth day.

4.3.2 Methods for Comparison

The main differences between above stated models for expertise estimation lie in two aspects:

1. Quality of answers. QLLM does not consider the quality of answers and it only estimates the similarities between the routed question q_r and each answerer's profile. Different from QLLM, The *Basic Model* and the *Smoothed Model* utilize the quality of answers in expertise estimation.

2. The way to estimate answer quality. The *Basic Model* estimates answer quality from the answerer's own profile, while the *Smoothed Model* leverages similar answerers' profiles to refine the estimation.

In total, we compare the QR performance of the following methods in our experiments:

- *QLLM*: For each question q_r in Set A, we use the QLLM to calculate the expertise on q_r for each answerer in Set B, and rank them according to their expertise. Thus, u_i's QR score (it is equal to expertise score here) for q_r is derived from:

$$QR(u_i, q_r, T) = P(q_r|q_{u_i}). \tag{4.17}$$

- *Basic Q*: For each question q_r in Set A, we use the *Basic Model* to calculate the answer quality, and apply Eq. (4.6) to estimate each answerer's expertise in Set B. We then rank them according to their expertise. Thus, u_i's QR score (it is equal to expertise score here) for q_r is derived from:

$$QR(u_i, q_r, T) = \alpha \cdot P(q_r|q_{u_i}) + (1-\alpha) \cdot Q_{BM}(u_i, q_r). \tag{4.18}$$

- *Smoothed Q*: For each question q_r in Set A, we use the *Smoothed Model* to calculate the answer quality, and apply Eq. (4.6) to estimate each answerer's expertise in Set B. We then rank them

according to their expertise. Thus, u_i's QR score (it is equal to expertise score here) for q_r is derived from:

$$QR(u_i, q_r, T) = \alpha \cdot P(q_r|q_{u_i}) + (1-\alpha) \cdot Q_{SM}(u_i, q_r). \quad (4.19)$$

- *QLLM+AE*: For each question q_r in Set A, we use the QLLM to calculate the expertise on q_r for each answerer in Set B, and estimate each answerer's availability within $T = 3$ days after the publish time of q_r. We then rank them according to their QR scores. Thus, u_i's QR score for q_r is derived from:

$$QR(u_i, q_r, T) = \gamma \cdot P(q_r|q_{u_i}) + (1-\gamma) \cdot A(u_i, T). \quad (4.20)$$

- *Basic Q+AE*: For each question q_r in Set A, we first use the *Basic Model* to calculate the answer quality and apply Eq. (4.6) to estimate each answerer's expertise in Set B, and then estimate each one's availability within $T = 3$ days after the publish time of q_r. Finally answerers are ranked according to their QR scores. Thus, u_i's QR score for q_r is derived from:

$$\begin{aligned} QR(u_i, q_r, T) &= \gamma \cdot [\alpha \cdot P(q_r|q_{u_i}) + (1-\alpha) \cdot Q_{BM}(u_i, q_r)] \\ &+ (1-\gamma) \cdot A(u_i, T). \end{aligned} \quad (4.21)$$

- *Smoothed Q+AE*: For each question q_r in Set A, we use the *Smoothed Model* to calculate the answer quality, and then apply Eq. (4.6) to estimate each answerer's expertise in Set B. After that, we estimate each one's availability within $T = 3$ days after the publish time of q_r. Finally answerers are ranked according to their QR scores. Thus, u_i's QR score for q_r is derived from:

$$\begin{aligned} QR(u_i, q_r, T) &= \gamma \cdot [\alpha \cdot P(q_r|q_{u_i}) + (1-\alpha) \cdot Q_{SM}(u_i, q_r)] \\ &+ (1-\gamma) \cdot A(u_i, T). \end{aligned} \quad (4.22)$$

4.3.3 Evaluation Metrics

The following evaluation metrics are adopted to evaluate the quality of generated QR List.

- *Precision at K* (*Prec@*K). For a set of new questions Q_r, Prec@K reports the fraction of successful QR when top-k answerers of the ranking list are returned. The criteria of a successful QR is defined as at least one answerer in the top-k of the ranking list actually answering the routed question. In this metric, positions of these answerers are not considered. The only key factor is whether there is at least one answerer in these K candidates who answered the routed question. Prec@K is calculated as:

$$Prec@K = \frac{\sum_{q_r \in Q_r} S(q_r, K)}{|Q_r|}, \tag{4.23}$$

$$S(q_r, K) = \begin{cases} 1, & \text{if QR for } q_r \text{ is successful;} \\ 0, & \text{otherwise.} \end{cases} \tag{4.24}$$

- *Mean Reciprocal Rank (MRR).* The reciprocal rank for an individual question q_r is the reciprocal of the rank at which the first answerer in QR list who actually answered q_r, or 0 if none of the answerers in QR list answered q_r. Here, the QR list is composed by all answerers. The MRR value for a set of questions Q_r is the mean value of each question's reciprocal rank. It is defined as:

$$MRR = \frac{1}{|Q_r|} \sum_{q_r \in Q_r} \frac{1}{Rank(q_r)}, \tag{4.25}$$

  where $Rank(q_r)$ is the rank of the first answerer who actually answered q_r in the QR list of q_r.

4.4 Experimental Results

In this section, we first show the result of QLLM, and discuss the rationality of routing new questions to those answerers who have answered similar questions previously. Then we present each method's performance in QR, and analyze the effects of parameter setting for expertise estimation and the impact of availability estimation for QR quality.

4.4.1 The Rationality of QR

Table 4.5 reports the number of successful QR as well as Prec@K using QLLM after testing 2,564 questions in Set A. We first note that even though we set K = ALL, i.e., we route the question q_r to all answerers in Set B, only 2/3 of questions will be answered. The reason is that the answerers for the remaining 851 questions do not have any record in Set B. Secondly, when $K = 1000$, more than half of the questions (with at least one answerer belonging to Set B) will be successfully routed to the answerers who actually answer this question. That means if we route the question to the top 6% $(1000/16,298)$ answerers who are familiar with the question, more than half of the questions will receive at least one answer. This observation demonstrates that it is reasonable and effective to route questions to those answerers who have answered similar questions previously. In our following experiments, we remove the 851 questions from Set A whose answerers do not appear in Set B, and use the remaining 1,713 ones as test questions.

Table 4.6 reports each method's performance measured by MRR. The value of α is 0.6 for those methods which consider answer quality. The value of β for Smoothed Q and Smoothed Q+AE is 0.8. Furthermore, we set $\gamma = 0.9$ for QLLM+AE, Basic Q+AE, and Smoothed Q+AE.

Table 4.5: Number of successful QR and Prec@K versus K for QLLM.

K	**5**	**10**	**20**	**40**	**60**	**80**	**100**	**1000**	**ALL (16,298)**
Number of Successful QR	80	121	175	253	312	350	386	876	1,713
$\mathbf{Prec@}K\mathbf{(\%)}$	3.12	4.72	6.83	9.87	12.17	13.65	15.05	34.17	66.81

Table 4.6: Different methods' MRR in QR.

QLL	**Basic Q**	**Smoothed Q**	**QLL+AE**	**Basic Q+AE**	**Smoothed Q+AE**
0.0389	0.0494	0.052	0.0405	0.0511	0.0541

4.4.2 Impact of Answer Quality

From Table 4.6 we observe that utilizing answer quality can improve the performance of QR significantly. The MRR values of Basic Q and Smoothed Q are 26.99% and 33.68% higher than that of QLLM, respectively. Similarly, the MRR values of Basic Q+AE and Smoothed Q+AE are 26.17% and 33.58% higher than that of QLLM+AE, respectively. These results demonstrate that utilizing answer quality can improve the accuracy in estimating answerer expertise on new question greatly. In order to explore the impact of α's value for QR performance, we fixed $\beta = 0.8$ and tested different settings of α. The results are reported in Figure 4.2.

First we observe that when $\alpha = 0.6$, both Basic Q and Smoothed Q attain the highest MRR. Furthermore, when $\alpha > 0.3$, the performance is always better than QLLM. With these findings, we believe that answer quality indeed provide great help for finding experts to the routed question.

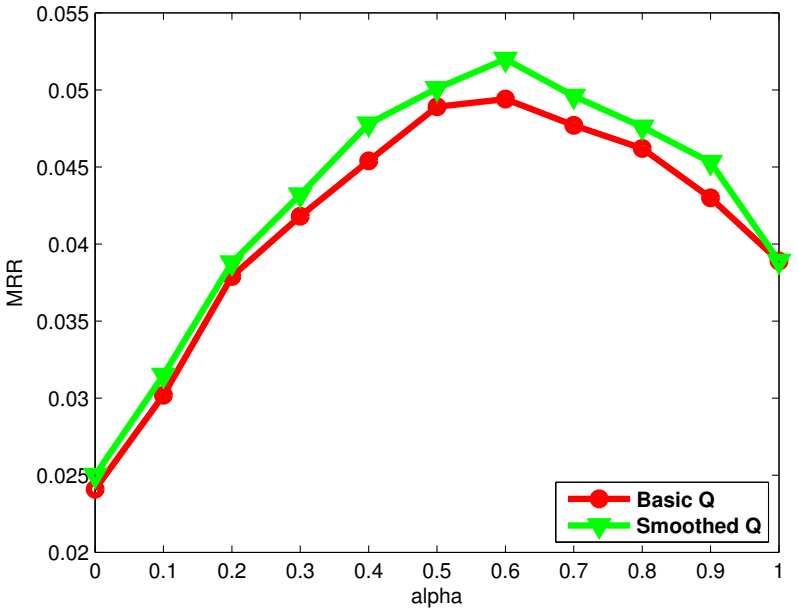

Figure 4.2: The MRR value of Basic Q and Smoothed Q versus various α.

4.4.3 Basic Q versus Smoothed Q

Smoothed Q outperforms Basic Q since the MRR of the former one is about 5% higher than that of the latter. We think this is due to the help of utilizing similar answerers' expertise on similar questions to smooth answer quality, especially for the answerers who answered few questions. Figure 4.3 gives the performance of Smoothed Q with different settings of β ($\alpha = 0.6$), from which we find that the value of β affects the QR quality of this method significantly. When $\alpha = 0$, which means we merely rely on similar answerers' expertise on similar questions for expertise estimation on routed question, the MRR is much lower than if there is no smoothing (i.e., $\alpha = 1$). The best performance of MRR is observed when $\beta = 0.8$.

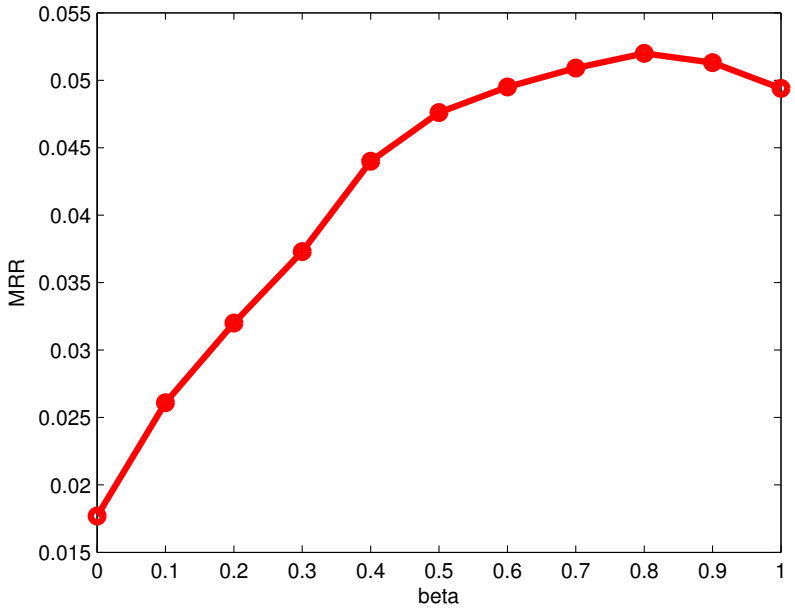

Figure 4.3: The MRR value of Smoothed Q versus various β.

4.4.4 Impact of Answerer Availability

Figure 4.4 presents each method's performance when considering answerer availability. First, routing questions based on answerer availability alone is very inaccurate: only about 1 out of 1,000 QRs will be successful when $\gamma = 0$. Second, QLLM+AE, Basic Q+AE, and Smoothed Q+AE perform the best when γ is around 0.9. When $\gamma = 0.9$, the MRR of these methods are 4.11%, 3.44%, and 4.04% higher than corresponding methods without availability estimation, respectively. Since answerers' answering history in our data set is incomplete, and the autoregressive model is imperfect (right now it can only deal with the prediction on each day), we believe the performance of QR can further be improved. Since availability estimation is helpful to improve QR performance, it is better to consider whether the answerer is available to provide answers in QR.

To sum up, our experiments demonstrate that it is reasonable and effective to route new questions to those answerers who have pre-

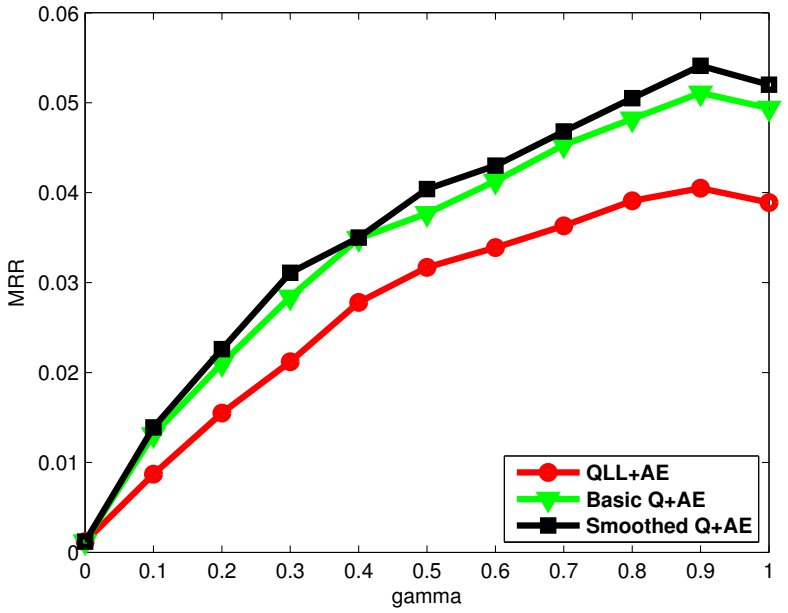

Figure 4.4: MRR versus γ across different methods.

viously answered similar questions. Utilizing answer quality in expertise estimation can improve the performance of QR significantly and using similar answerers' answer quality on similar questions is helpful in calculating answer quality for new questions. In addition, availability estimation further boosts the performance of QR.

The best MRR value in our methods is 0.0541, which means on average each tested question will get at least one answer if we route it to the top 19 ranked answerers after answerer ranking. Considering there is a total of 16,298 answerers for ranking, the result is promising.

4.5 Summary

In this chapter, we propose a framework for *Question Routing* in CQA services. QR is a process that routes newly posted questions to a set of answerers who are most likely to give answers in a short

period of time. The proposed QR framework considers both answerer expertise and answerer availability for providing answers in a range of time. We conduct experiments on a collection of data from Yahoo! Answers. The results show that it is effective to route questions to answerers who have answered similar questions previously. It also demonstrates that leveraging answer quality can greatly improve the performance of QR. In addition, utilizing similar answerers' answer quality on similar questions provides more accurate expertise estimation, and thus obtains better QR performance. Furthermore, availability estimation improves the performance of QR. Based on the experimental results so far, we believe that our QR framework has the ability to route new questions to those answerers who will provide answers in a short period of time.

In the future, further study is needed to investigate the performance of our approach across different CQA portals, as well as different question domains (instead of the *Computer & Internet* category). In addition, we plan to explore more advanced availability estimation models.

□ **End of chapter.**

Chapter 5

Category-sensitive Question Routing

5.1 Problem and Motivation

In the last chapter, we propose the question routing (QR) framework which aims to route newly posted questions to potential answerers. The appropriateness of potential answerers is evaluated based on archives of their previously answered questions. In the framework, answerer expertise and answerer availability are utilized to rank potential answerers, with top-k being chosen as eligible ones. Therefore, expertise estimation plays a key role in QR. Volumes of studies have been conducted regarding expertise estimation, including Query Likelihood Language Model (QLLM) [84], Cluster-based Language Model (CBLM) [167], mixture of Latent Dirichlet Allocation (LDA) and QLLM [80]. However, for *all answerers*, the expertise has been estimated for QR, even to answerers without any experience of routed questions. For *an answerer*, a complete set of questions the answerer has answered is utilized in the models, although a certain amount of answered questions might be irrelevant to the questions to be routed.

To address these two problems, in this chapter we focus on utilizing category information in QR. As shown in Figure 5.1, each question is assigned with a particular category in CQA services. In the example, the question belongs to the leaf category *Monitors* and the top category *Computers & Internet*. The category information

Figure 5.1: An example of question category in CQA services (captured from Yahoo! Answers on January 20, 2011).

of new questions would allow much latitude in filtering irrelevant answerers among *all answerers*, together with screening irrelevant questions of *an answerer* to enhance the efficiency of expertise estimation. Based on this idea, we construct category-answerer indexes for filtering irrelevant answerers, and develop category-sensitive LMs for estimating answerer expertise. Figure 5.2 shows the incorporation of category information for expertise estimation.

This chapter is organized as follows. Category-answerer indexes and category-sensitive LMs are developed in Section 5.3. Experimental setup as well as results are then reported and discussed in Section 5.4 and Section 5.5. A summary is given in Section 5.6.

5.2 Question Category for Routing Questions

5.2.1 Category-Answerer Indexes

In the area of expert finding [6, 60], irrelevant document authors are sifted out to find the most appropriate experts for the topic. Similarly, filtering irrelevant answerers is beneficial in finding the most relevant answerers for answering questions regarding certain topics. Therefore, we construct category-answerer indexes to filter irrele-

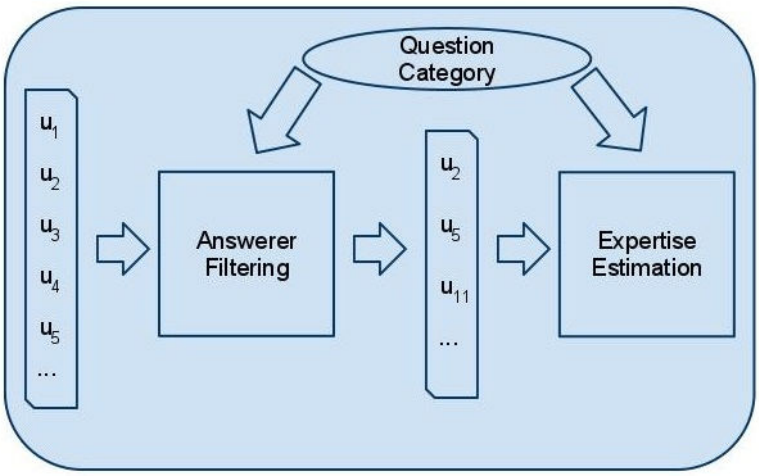

Figure 5.2: Question category for expertise estimation in QR.

vant answerers for expertise estimation. In our study, we establish two indexes:

1. *Severe index*: Only those answerers who have answered at least one question in the routed question's leaf category are kept, with the remaining answerers being sifted out.

2. *Lenient index*: Answerers who have answered at least one question in the routed question's top category are indexed.

The application of the category-answerer indexes filters out a great number of irrelevant answerers, with only relevant answerers being kept, thus reducing the number of answerers for expertise estimation. Subsequently, we apply the two category-sensitive LMs presented in the following to estimate potential answerers' expertise for routed questions.

5.2.2 Category-sensitive Language Models

Language models (LMs) are originally used in the community of speech recognition [151], before being introduced to information re-

trieval (IR) [48, 118]. The idea of LMs in IR is to estimate the likelihood of generating a query from the document. As and basic LM in IR is QLLM [118] (details about QLLM are presented in Chapter 4), our category-sensitive LMs extend it through incorporating category information.

QLLM assumes that an answerer will be of expertise in answering the new question q_r if the probability of generating q_r from his profile is high. In addition, it utilizes a complete profile including all previously answered questions for estimation. However, the whole answerer profile, as discussed, might include a certain amount of answered questions under a wide range of categories, with little relevance to q_r. The content of irrelevant questions will thus jeopardize the quality of expertise estimation. Consider an example:

> Alex, a senior Java programmer, is an active answerer in Yahoo! Answers. He has answered more than 1,000 questions in terms of Java programming as well as 100 questions about Java coffee.
>
> Bob, a cafe manager, is also a frequent user of Yahoo! Answers. He answered around 300 questions about Java coffee, but he knows little about Java programming.
>
> Carl, a college student, now asks a question "I met a problem in making Java, any ideas" in "*Food & Drink*" category.

For the above example, the QLLM would predict Alex to be a better expert than Bob, as Alex's profile includes more "Java" linguistic texts (1,100 questions in total) than those of Bob (300 questions), although Bob might be more appropriate to answer Carl's question. To address the above problem, we propose two novel LMs by incorporating question category in QLLM (that's why we call them "category-sensitive" LMs). In the following, we present these two algorithms in turn.

BCS-LM. In CQA portals, questions are equipped with hierarchical and systematic categories. When an asker posts a ques-

tion, he is required to choose a (leaf) category that the question belongs to (see Figure 5.1). Formally, let q_{u_i} denote all previously answered questions by answerer u_i (i.e., u_i's profile), and $C = \{c_1, c_2, c_3, ..., c_n\}$ represents all leaf categories, then for a new question q_r in c_j, u_i's expertise on q_r, which is represented by $E(u_i, q_r, c_j)$ is defined as follows:

$$
\begin{aligned}
E(u_i, q_r, c_j) &\equiv P_{bcs}(u_i|q_r, c_j), && (5.1)\\
P_{bcs}(u_i|q_r, c_j) &\propto P_{bcs}(q_r, c_j|u_i)P(u_i), && (5.2)\\
P_{bcs}(q_r, c_j|u_i) &= P_{bcs}(q_r|c_j, u_i)P(c_j|u_i), && (5.3)\\
P_{bcs}(q_r|c_j, u_i) &= P_{bcs}(q_r|c_j, q_{u_i}) = \prod_{\omega \in q_r} P(\omega|q_{u_{ij}}), && (5.4)
\end{aligned}
$$

and

$$
P(\omega|q_{u_{ij}}) = (1-\lambda)P_{ml}(\omega|q_{u_{ij}}) + \lambda P_{ml}(\omega|Coll), \qquad (5.5)
$$

where c_j is q_r's category, $P(c_j|u_i)$ denotes the probability of answering questions in c_j for u_i, and $q_{u_{ij}}$ represents the question texts of all previously answered questions in c_j for u_i. We call this model as the basic category-sensitive LM (BCS-LM).

Turn to the aforementioned example, the estimation of Alex and Bob's expertise to Carl's question will be adjusted accordingly, using the category-sensitive QLLM.

TCS-LM. It is also noted that BCS-LM is based on the *same-leaf-category* assumption, with potential answerers under similar leaf categories being omitted. CQA portals like Yahoo! Answers usually provide category hierarchies to organize questions. Under one main category, there exist similar leaf categories, e.g., the leaf categories of *Programming & Design* and *Software*. Answerers with expertise in *Programming & Design* may also be experts on questions asked in *Software*. Based on this idea, we propose the trans-

ferred category-sensitive QLLM (TCS-LM) as follows:

$$P_{tcs}(q_r, c_j|u_i) = \frac{\beta P_{bcs}(q_r, c_j|u_i) + \sum_{c_k \in Tran(c_j)} T(c_k \to c_j) P_{bcs}(q_r, c_k|u_i)}{\beta + \sum_{c_k \in Tran(c_j)} T(c_k \to c_j)}, \tag{5.6}$$

where β adjusts the weight between the original leaf category and other similar leaf categories. Lower β value provides more weight to similar categories, and vice versa. $Tran(c_j)$ denotes the set of categories which are transferable from (similar to) c_j, and $T(c_k \to c_j)$ represents the probability of transfer from category c_k to c_j. Furthermore, we define

$$c_k \in Tran(c_j) \text{ if } T(c_k \to c_j) \geq \delta, \tag{5.7}$$

where δ is a threshold between 0 and 1.

To estimate the transferring probability between two categories, we attempt two approaches: answerer-based approach and content-based approach.

Answerer-based method assumes that if there are many same answerers posting answers in two categories, these two categories should be similar with each other. It constructs a category-answerer matrix E from resolved questions (see Figure 5.3), with each row of E representing one answerer and each column representing one (leaf) category. The value of e_{ij} denotes the number of answers u_i provided in category c_j. Let $\mathbf{e}_j$ and $\mathbf{e}_k$ represent two column vectors of c_j and c_k, the transferring probability ($T_{ans}(\cdot)$) between c_j and c_k is estimated from their cosine similarity:

$$T_{ans}(c_j \to c_k) = T_{ans}(c_k \to c_j) = \frac{\mathbf{e}_j \cdot \mathbf{e}_k}{|\mathbf{e}_j||\mathbf{e}_k|}. \tag{5.8}$$

Similarly, the content-based method assumes that if there are many identical terms appearing in two categories' questions, these two categories should be similar to each other. It constructs a category-term

$$E = \begin{bmatrix} 2 & 0 & 3 & 4 & 0 \\ 0 & 3 & 0 & 0 & 0 \\ 4 & 8 & 4 & 5 & 1 \\ 7 & 0 & 6 & 0 & 2 \\ 0 & 4 & 5 & 6 & 1 \end{bmatrix}$$

Category (columns) / Answerer (rows); $\mathbf{e}_j$, $\mathbf{e}_k$

Figure 5.3: An example of Matrix E.

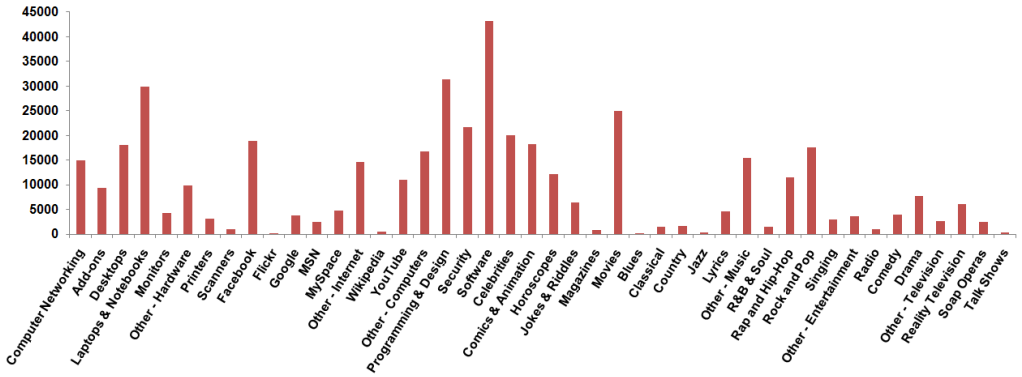

Figure 5.4: Question distribution across different leaf categories (The first 20 are leaf categories under *Computers & Internet*, the left are leaf categories under *Entertainment & Music*).

matrix from resolved questions and also utilizes the cosine similarity as the transferring probability between two categories.

5.3 Experimental Setup

To examine the QR efficiency of applying question category, we conduct experiments on category-answerer indexes and category-sensitive LMs. In this section, we first present the data collection procedures, and then describe the compared methods with our category-sensitive LMs. In the following, evaluation metrics are presented. At last, we detail the model training for some algorithms.

Table 5.1: Description of our data set (after stop words removing and stemming).

Item	Value
Number of questions	433,072
Number of answers	1,510,531
Average number of answers for one question	3.49
Maximum number of answers for one question	50
Mean first reply duration (in minutes)	197.32
Average question length in words	43.87
Average answer length in words	30.08
Number of askers	240,277
Number of answerers	270,043
Number of both askers and answerers	68,551
Number of askers	171,726
Number of answerers	201,492

5.3.1 Data Collection

The data consist of over 400 thousands resolved questions, which are crawled from June 2010 to October 2010 under *Computers & Internet* and *Entertainment & Music* categories of Yahoo! Answers through Yahoo! Answers API[1]. Under the two categories, there are 20 and 25 leaf categories, respectively[2]. Figure 5.4 presents the distribution of questions over leaf categories, and Table 5.1 reports the statistics of our data set.

It is worth noting that our experiments mainly employ question texts for answerer profiling[3]. Answer texts are intentionally ex-

[1]http://developer.yahoo.com/answers/

[2]We manually remove questions in *Polls & Surveys* because questions in this leaf category are seeking for a large group of user opinions or suggestions while not require a few number of expert answers as questions in other categories. Therefore, our data set includes 44 leaf categories.

[3]For the comparison purpose, the chapter also attempts the Cluster-based LM (CBLM) [167] to estimate answerer expertise. CBLM resorts to answer texts for estimating the contribution of an answerer to that particular question and the respective cluster. See more discussions in Section 5.3.2.

cluded from an answerer profile based on two considerations. First, using question texts alone in answerer profile is found to perform more consistently across various data sets and achieve more satisfactory empirical results, comparing with those involving both question and answer texts (see Liu et al. [84]). In this connection, only question texts are selected to most algorithms in the experiment. Second, including answer texts is not efficient in terms of two aspects. On the one hand, similarities between texts of question A and those of question B could not ensure that texts of answers to question A are also similar to texts of question B. Specifically, consider the following example:

> *Question A*: What are your favorite movies?
> *Question B*: What are your top ten favorite movies of all time?
> *An Answer to Question A*: Avatar and The Blind Side.

In the above example, no text similarities are observed between texts of the answer to question A and texts of question B. On the other hand, answer texts usually contain redundant or irrelevant texts, which might hinder the efficiency of LMs.

Therefore, as for the crawled resolved questions, the information of each question comprises:

1. Affiliated category chosen by the original asker;
2. Texts of question (subject and content), with stop words being excluded and words being stemmed;
3. Answerers' IDs of each question.

Furthermore, all questions are classified into two sets:

- Set A (Test data): questions posted after 6 May, 2010 used as questions to be routed;
- Set B (Archive data): the remaining questions.

As such, Set A is made up of 382,695 questions, 1,335,892 answers and 243,167 answerers. Set B is composed of 50,377 questions, 174,639 answers and 49,466 answerers. In addition, we set two ground truths:

1. GT-A: Answerers who answered questions in Set A;
2. GT-BA: Answerers who give the best answers for questions in Set A.

GT-BA is more rigorous since this standard only allows one "relevant" answerer (the one who gives the best answer) for each question while GT-A treats all answerers as "relevant" ones.

5.3.2 Methods for Comparison

For answerer filtering, we compare the effect of using the severe index and the lenient index to no index used. For expertise estimation, we choose the cluster-based language model (CBLM) and the mixture of LDA and QLLM (LDALM) to compare with category-sensitive LMs based on the following two considerations:

1. In CBLM [167], similar questions under the same topic are clustered, and answerer expertise is estimated through calculating the answerer's contribution to each cluster as well as the similarity between the routed question and each cluster. In CQA portals, each leaf category could be treated as a cluster, and CBLM is thus employed. Therefore, we implement CBLM to explore whether such "category-sensitive" setting is comparable with our category-sensitive LMs.
2. Liu et al. [80] show that utilizing latent topics boosts the performance of QLLM for expertise estimation. Therefore, we intend to compare the effectiveness of latent topics to explicit categories, as they both consider semantics in expertise estimation.

The two models of CBLM and LDALM are briefly introduced in the following.

1. *CBLM*. Each answerer's expertise on the routed question q_r is estimated from his contribution to each cluster and the similarity between each cluster and q_r:

$$P(q_r|u_i) = \sum_{Cluster} \prod_{\omega \in q_r} P(\omega|\theta_{Cluster})^{n(\omega,q_r)} con(Cluster, u), \tag{5.9}$$

In the above equation,

$$\begin{aligned} P(\omega|\theta_{Cluster}) &= (1-\lambda)P(\omega|Cluster) \\ &+ \lambda P(\omega|Coll), && (5.10) \\ con(Cluster, u_i) &= \sum_{qa} con(qa, u_i), && (5.11) \\ con(qa, u_i) &= \frac{\prod_{\omega \in q} P(\omega|\theta_{a_{u_i}})}{\sum_{(qa)'} \prod_{\omega \in q'} P(\omega|\theta_{a'_{u_i}})}, && (5.12) \\ P(\omega|\theta_{a_u}) &= (1-\lambda)P(\omega|a_u) \\ &+ \lambda P(\omega|Coll), && (5.13) \end{aligned}$$

where ω represents a term, $n(\omega, q_r)$ means the number of times that ω appears in q_r, $\theta_{Cluster}$ denotes the language model of the cluster, and $P(\omega|\theta_{Cluster})$ denotes the maximum likelihood estimation of ω in all question texts in that cluster. Similarly, $\theta_{a_{u_i}}$ represents the language model of u_i's answers (a_{u_i}). In addition, $con(qa, u_i)$ means u_i's contribution to one question-answer pair in which u_i gives the answer a, $\lambda \in [0, 1]$ is a weighting parameter, and $Coll$ denotes the whole collection. Note that extra information (i.e., answer texts) is required to calculate answerers' contributions to each cluster. In our experiments, we treat each leaf category as a cluster.

2. *LDALM*. Each answerer's expertise on the routed question q_r is estimated from both QLLM and LDA. LDA captures answerer expertise on latent topics together with the relationships

between latent topics and q_r, which bridges the lexical gap in QLLM:

$$P(q_r|u_i) = \prod_{\omega \in q_r} P(\omega|\theta_{u_i})^{n(\omega,q_r)}, \quad (5.14)$$

$$P(\omega|\theta_{u_i}) = \delta P_{LDA}(\omega|\hat{\theta}, \hat{\phi}, \theta_{u_i}) + (1-\delta) P_{QLLM}(\omega|\theta_{u_i}), \quad (5.15)$$

$$P_{LDA}(\omega|\hat{\theta}, \hat{\phi}, \theta_{u_i}) = \sum_{z=1}^{Z} P(\omega|z, \hat{\phi}) P(z|\hat{\theta}, \theta_{u_i}), \quad (5.16)$$

where θ_{u_i} denotes the language model of u_i's profile, δ is a weighting parameter, Z is the number of latent topics, $\hat{\theta}$ and $\hat{\phi}$ are the posterior estimates of parameters in the generation process of LDA. $P_{QLLM}(\omega|\theta_{u_i})$ can be estimated using Eq. (**??**). We set $Z = 100$ and $\delta = 0.5$ empirically according to [80].

In addition, the original QLLM is included in comparisons as the baseline method. Together with the proposed two category-sensitive LMs, we compare the performance of expertise estimation with five methods altogether.

All of these methods include the user prior $P(u_i)$, which represents the probability of routing the new question to u_i without knowing anything of u_i. Liu et al. [84] assume $P(u_i)$ subject to a uniform distribution. Suppose the number of answerers is N,

$$P_{Uniform}(u_i) = \frac{1}{N}. \quad (5.17)$$

However, when we build each answerer's profile, we already know some information of u_i, such as the ask-answer links among users and the number of answers for each answerer. Zhou et al. [167] model $P(u_i)$ as the authority of u_i among all answerers and applies PageRank [15] in calculation. However, PageRank is not appropriate in representing answerers' authorities well because the whole graph is not connected, and authorities in connected sub-graphs (rather

than in the whole graph) are estimated. Bouguessa et at. [14] demonstrate that in-degree (total number of answers) is more effective in CQA services. Therefore, we normalize in-degree for each answerer and model $P(u_i)$ as the number of answers u_i provided ($N_{ans}(u_i)$) divided by the total number of answers, so

$$P_{Indegree}(u_i) = \frac{N_{ans}(u_i)}{M}, \tag{5.18}$$

where M denotes the total number of previously answered answers.

5.3.3 Evaluation Metrics

We use the average number of potential answerers per routed question and loss of recall to measure the effect of category-answerer indexes for answerer filtering. The average number of potential answerers per question measures the influence on filtering out irrelevant answerers, and the loss of recall measures the degree of relevant answerers that are eliminated.

We adopt the Precision at K (Prec@K), the Mean Average Precision (MAP), the Mean Reciprocal Rank (MRR), and the Mean QR Time (MQRT) as evaluation metrics for various LMs in expertise estimation, as they are widely used in evaluating the quality of retrieval results [96]. We have presented the definitions of Prec@K and MRR in the last chapter. In the following, we describe MAP and MQRT one after another.

- *Mean Average Precision*: For a set of new questions Q_r, MAP measures the mean of the average precision for each question

Table 5.2: Transferred probabilities between partial leaf categories (answerer-based method).

From \ To	Software	Printers	Comedy	Lyrics
Programming & Design	0.2975	0.0251	0.0026	0.0026
Scanners	0.1158	0.5604	0.0014	0.0008
Drama	0.0053	0.0006	0.2593	0.0137
Other - Music	0.0102	0.0019	0.0273	0.1683

q_r in QR:

$$MAP = \frac{\sum_{q_r \in Q_r} AvgP(q_r)}{|Q_r|}, \tag{5.19}$$

$$AvgP(q_r) = \frac{\sum_{k=1}^{N_r}(P_r(k) \cdot IsAns(k))}{NRA_r}, \tag{5.20}$$

$$P_r(k) = \frac{NRA_r(k)}{k}, \tag{5.21}$$

where Q_r is a set of questions to be routed, N_r is the number of potential answerers for q_r generated from answerer filtering, NRA_r is the number of real answerers for q_r, $IsAns(k)$ is a binary function denoting whether the k^{th} answerer actually answered q_r, and $NRA_r(k)$ denotes the number of real answerers in the top-k answerers for q_r.

- *Mean QR Time*: It calculates the average time spent on routing (including answerer filtering, expertise estimation, and answerer ranking) one question as the metric of time efficiency for all methods.

Table 5.3: Transferred probabilities between partial leaf categories (content-based method).

From \ To	Software	Printers	Comedy	Lyrics
Programming & Design	0.2250	0.0236	0.0116	0.0116
Scanners	0.1676	0.2671	0.0049	0.0034
Drama	0.0136	0.0020	0.5481	0.0376
Other - Music	0.0443	0.0070	0.0748	0.2922

5.3.4 Model Training and Parameter Setting

The transferred probability from one leaf category to another is calculated in advance. Table 5.2 and Table 5.3 report partial results using answerer-based and content-based methods respectively, from which we find that the leaf categories under same high-level category (e.g., *Programming & Design* $\rightarrow$ *Software*) are more transferable than those under different ones (e.g., *Programming & Design* $\rightarrow$ *Comedy*). Comparing the results of answerer-based method and those of content-based method, the latter generate higher similarities between irrelevant categories, e.g., *Programming & Design* $\rightarrow$ *Comedy*, *Other - Music* $\rightarrow$ *Software*, etc. The influence of these two approaches on TCS-LM will be detailed in Section 5.4.2.

We employ the tool GibbsLDA++[4] to estimate the posterior probabilities of LDA (say, θ of each answerer and ϕ of each topic). The default setting is adopted, and the number of latent topics is set to 100. We set $\beta = 3.5$ for TCS-LM in the experiments empirically as this setting yields the best performance. Because it is time-consuming to test all questions in Set A (Test set), we sample 440 questions randomly from Set A (10 questions from each leaf category) as testing data. All algorithms are tested on a PC equipped

[4]http://gibbslda.sourceforge.net/

Table 5.4: Effects of using category-answerer indexes on answerer filtering.

Type	Average number of potential answerers		Loss of recall
	Before filtering	After filtering	
Severe	243,167	19,235 (↓92.09%)	0.24
Lenient	243,167	137,171 (↓43.59%)	0.14

with two 2.4GHz CPUs and 3GB RAM.

5.4 Experimental Results

This section reports and discusses experimental results of using category-answerer indexes to shorten the list of potential answerers and applying category-sensitive LMs to expertise estimation.

5.4.1 Category-Answerer Indexes

Table 5.4 reports the effect of answerer filtering using two different indexes. The severe index reduces the average number of potential answerers by 92%, while lenient index reduces 43% of candidates, with a significant decrease of computing costs being identified. In the recall, among all 50,377 test questions, there are 38,343 questions satisfying the condition that at least one answerer is indexed by the severe index (recall of 76%). In addition, there are 43,235 questions whose answerers are indexed when lenient indexes are applied (recall of 86%). Due to the limitation of sample size, the improvement of recall may be obtained in a real world situation. So far, experimental results have supported the inclusion of leveraging question category as a filter to omit irrelevant answerers, which makes QR more efficient with only a small loss of recall. Since the severe index filters more irrelevant answerers with little decrease of recall, we employ the severe index in the following experiments.

Table 5.5: Different methods' Prec@K in QR versus various Ks using GT-A (best results are shown in bold).

K	QLLM	BCS-LM	TCS-LM	LDALM	CBLM
1	0.0795	0.1114 (↑40.13%)	**0.1227** (↑54.34%)	0.0989 (↑24.40%)	0.0000
3	0.1659	**0.2364** (↑42.50%)	0.2340 (↑41.05%)	0.1950 (↑17.54%)	0.0000
5	0.2091	**0.2727** (↑30.42%)	0.2705 (↑29.36%)	0.2455 (↑17.41%)	0.0000
10	0.2705	0.3386 (↑25.18%)	**0.3455** (↑27.73%)	0.3102 (↑14.68%)	0.0000
20	0.3386	0.3909 (↑15.45%)	**0.3932** (↑16.13%)	0.3710 (↑9.57%)	0.0091
40	0.4136	0.4523 (↑9.36%)	**0.4591** (↑11.00%)	0.4392 (↑6.19%)	0.0273
60	0.4477	**0.4818** (↑7.62%)	0.4795 (↑7.10%)	0.4649 (↑3.84%)	0.0545
80	0.4727	**0.4955** (↑4.82%)	0.4909 (↑3.85%)	0.4867 (↑2.96%)	0.0727
100	0.4909	**0.5159** (↑5.09%)	0.5114 (↑4.18%)	0.4979 (↑1.43%)	0.0795

5.4.2 Category-sensitive Language Models

Table 5.5 and Table 5.6 report Prec@K for all algorithms with different Ks from 1 to 100 under GT-A and GT-BA. Table 5.7 gives the time-efficiency of each method in QR based on MQRT. Moreover, Table 5.8 and Table 5.9 present the MRR and MAP of all methods[5]. It is worth noting that the ranking algorithm is not tuned in our experiments, and all potential answerers are ranked for each test question. Therefore, it is expected that the time for all methods will be further reduced with advanced top-k algorithms.

Higher accuracies. From Table 5.5 we observe that, for various of Ks, both BCS-LM and TCS-LM outperform QLLM significantly on Prec@K. For instance, when routing questions to the top 1 answerers, on average QLLM gives less than 8 successful routings per 100; BCS-LM and TCS-LM produce more than 11 and 12 successful routings, which improve QLLM by 40.13% and 54.34%, respectively. For other Ks, category-sensitive LMs also perform better than QLLM.

The MRR of BCS-LM and TCS-LM increase that of QLLM by

[5]Under GT-BA, MRR is equals to MAP.

Table 5.6: Different methods' Prec@K in QR versus various Ks using GT-BA (best results are shown in bold).

K	QLLM	BCS-LM	TCS-LM	LDALM	CBLM
1	0.0568	0.0682 (↑20.07%)	**0.0773** (↑36.09%)	0.0668 (↑17.61%)	0.0000
3	0.1091	**0.1477** (↑35.38%)	0.1409 (↑29.15%)	0.1258 (↑15.31%)	0.0000
5	0.1363	**0.1705** (↑25.09%)	0.1659 (↑21.72%)	0.1655 (↑21.42%)	0.0000
10	0.1705	0.2068 (↑21.29%)	**0.2091** (↑22.58%)	0.1950 (↑14.40%)	0.0000
20	0.2205	**0.2591** (↑17.51%)	0.2523 (↑14.42%)	0.2472 (↑12.11%)	0.0023
40	0.2750	0.3114 (↑13.24%)	**0.3136** (↑14.04%)	0.2891 (↑5.13%)	0.0091
60	0.3023	**0.3386** (↑12.01%)	**0.3386** (↑12.01%)	0.3109 (↑2.84%)	0.0295
80	0.3182	0.3432 (↑7.86%)	**0.3455** (↑8.58%)	0.3225 (↑1.35%)	0.0386
100	0.3364	**0.3614** (↑7.43%)	0.3591 (↑6.75%)	0.3365	0.0386

29.66% and 34.59%. From the definition of MRR, each new question will be answered by at least one answerer in the top 5 answerers using BCS-LM or TCS-LM. However, with QLLM, on average we have to route the question to top 7 answerers to get an answer. As to MAP, BCS-LM and TCS-LM improve QLLM by 33.08% and 37.29%, respectively. Therefore, it shows that category-sensitive LMs give more accurate rankings on the whole.

Table 5.6 reports similar results, although each method's performance is decreased. To sum up, the above results have assured the effectiveness of utilizing category information in expertise estimation.

Lower costs. Now let's turn to the time costs of QLLM and category-sensitive LMs. Table 5.7 gives the average time of routing a question for each model. We find that BCS-LM saves 47.16% of time, and TCS-LM costs 13.80% less time than QLLM, which demonstrates that category-sensitive LMs are more time-efficient than QLLM in expertise estimation and thus make QR faster. The lower costs of BCS-LM lie in that only relevant profiles are utilized in expertise estimation, which reduces computational costs.

Table 5.7: Different methods' MQRT in QR (in seconds).

QLLM	BCS-LM	TCS-LM	LDALM	CBLM
10.4271	5.5098	8.9884	16.7689	4.2488

The time complexity of QLLM is linear to the number of answerers since all probabilities can be pre-computed. Let A denote the set of answerers, and A_{q_c} represent the set of answerers in the routed question q' leaf category, the time complexity of QLLM and BCS-LM would be $O(|A|)$ and $O(|A_{q_c}|)$, respectively. Because $|A_{q_c}| \ll |A|$, BCS-LM reduces computational costs significantly. TCS-LM spends more time than BCS-LM because of employing profiles in relevant categories for expertise estimation. Although TCS-LM is more time-consuming than BCS-LM, it is possible to reduce the time through parallel computing since the expertise estimation with different categories' profiles is independent from each other.

BCS-LM versus TCS-LM. Looking at Table 5.5, we find that utilizing similar categories improve accuracies of expertise estimation when K is small. In particular, the Prec@1 of TCS-LM is 10.14% higher than that of BCS-LM. In addition, the Prec@10 of TCS-LM is 2.04% more accurate than that of BCS-LM. Although when K becomes large (say, higher than 40), TCS-LM improves less or even slightly worse than BCS-LM. The former one is still a better choice as a QR system must route a question to minimum number of potential answerers in practice. Table 5.8 indicates that MRR and MAP of TCS-LM are also better than those of BCS-LM. When GT-BA is applied, similar results are found (see Table 5.6 and Table 5.9). TCS-LM utilizes similar categories' profiles and assigns weights to these profiles according to the degree of similarities. Therefore, they give more precise expertise estimation and thus improve QR performance.

Impact of transferring probability. Noting that we use two

Table 5.8: MRR and MAP of various models under GT-A (best results are shown in bold).

Method	MRR	MAP
QLLM	0.1460	0.1070
BCS-QLLM	0.1893 (↑29.66%)	0.1424 (↑33.08%)
TCS-QLLM	**0.1965** (↑34.59%)	**0.1469** (↑37.29%)
LDALM	0.1695 (↑16.10%)	0.1281 (↑19.72%)
CBLM	0.0031	0.0024

approaches to calculate transferring probabilities: one is answerer-based, and the other is content-based. Figure 5.5 presents the MRR of TCS-LM with various δ using the answerer-based and the content-based approaches when GT-A is used as the groundtruth. We also plot the MRR of BCS-LM as a baseline. From Figure 5.5, we find the answerer-based method performs better than the content-based method. As noted in Section 5.3.4, the content-based method gives higher similarities between irrelevant categories. We conjecture that different categories still share some common words although stopwords have been removed. However, the answerer-based method avoids term-level computation, and leverages community information which is more suitable for estimating categories' similarities. In real world, answerers prefer to answer questions in relevant-with-each-other categories.

Category-sensitive LMs versus CBLM versus LDALM. Both CBLM and LDALM are the extensions of the original QLLM. The former one introduces clusters into QLLM, and groups questions according to similar topics (in our experiments each leaf category was treated as a cluster). The latter integrates QLLM with LDA, which considers answerer expertise on latent topics together with the relationships between latent topics and routed questions. We compare category-sensitive LMs with CBLM to explore how to better utilize

Table 5.9: MRR and MAP of various models under GT-BA (best results are shown in bold).

Method	MRR	MAP
QLLM	0.0976	0.0976
BCS-QLLM	0.1178 (↑20.70%)	0.1178 (↑20.70%)
TCS-QLLM	**0.1219** (↑24.90%)	**0.1219** (↑24.90%)
LDALM	0.1133	0.1133
CBLM	0.0015 (↑16.09%)	0.0015 (↑16.09%)

category information in QR, and with LDALM to investigate the effectiveness of category and latent topics.

Among these four methods, CBLM performs the worst. A possible reason is that a large amount of answerers only answered in one cluster (leaf category), as such their contributions to this cluster are one according to Eq. (5.9). Under this circumstance, these answerers' expertise is actually measured by those clusters' "expertise", which will cause a great number of answerers to own the same expertise and thus render the ranking meaningless. LDALM increases Prec@K of QLLM, which shows the impact of utilizing latent topics, but the explicit question category provides more help than latent topics because category-sensitive LMs outperform LDALM at various Ks. MRR and MAP of these four methods report the similar results (detail will not be provided here).

When turning to MQRT, we find that CBLM works the best, followed by BCS-LM and TCS-LM, while LDALM costs much more time in inference. CBLM estimates answerer expertise by combining an answerer's contribution to each cluster (which is precomputed) with the probability of generating the routed question from each cluster (which is efficient to calculate), thus it makes the fastest estimation. However, the estimation made by CBLM is the most inaccurate, as stated above. On the whole, category-sensitive

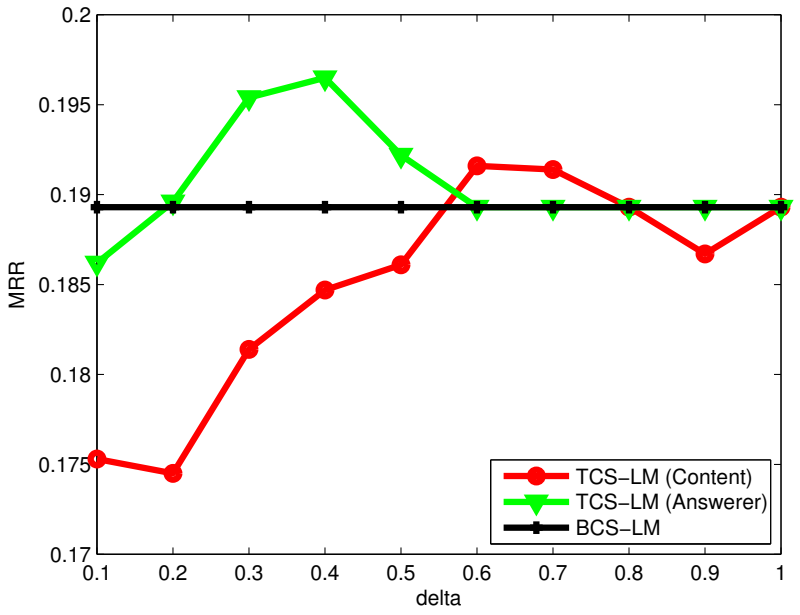

Figure 5.5: MRR for TCS-LM using answerer-based and content-based approaches to estimate transferring probability under GT-A.

LMs are the most time-efficient among the four methods.

In summary, category-sensitive LMs give more accurate expertise estimation than CBLM and LDALM and at the same time keep high time-efficiency.

User prior. To explore different user priors' influence on QR, we apply each of them to all expertise estimation algorithms. The Prec@K of LM and BCS-LM with different answerer priors are reported in Figure 5.6 and Figure 5.7. We find that with more prior information, the performance of QR is improved significantly: both methods' Prec@K values are steadily boosted from using $P_{Uniform}(u)$ to using $P_{Indegree}(u)$ as answerer prior, regardless of whether GT-A or GT-BA is used as the ground truth. Other methods report similar results. Thus, answerers are suggested to be treated differently according to their answering history.

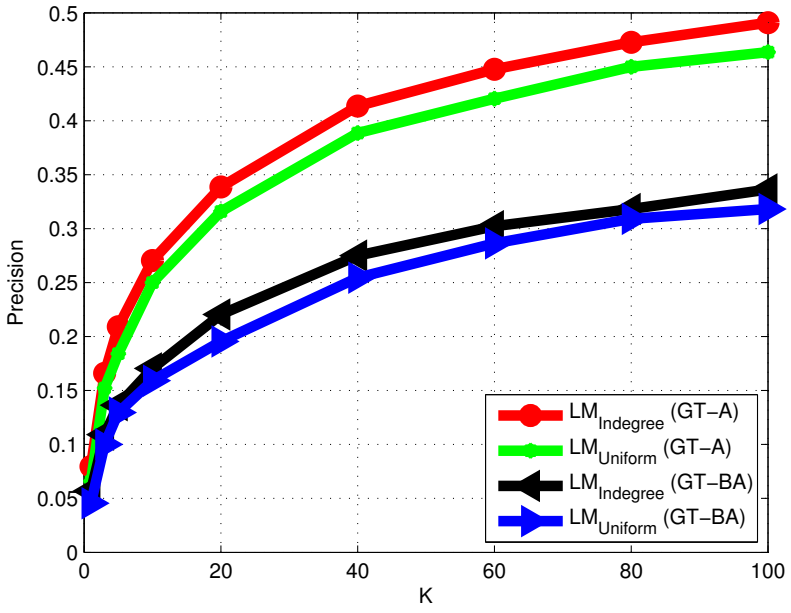

Figure 5.6: Prec@K of LM with different answerer priors.

5.4.3 Discussion

Results of the experiments on category-answerer indexes and category-sensitive LMs demonstrate that: (1) for each question to be routed, category-answerer indexes successfully filter irrelevant answerers to produce a shorter list of relevant answerers for expertise estimation; and (2) for each of those filtered relevant answerers, category-sensitive LMs obtain more accurate question routings, relative to traditional QLLM and two state-of-the-arts models.

Results of the experiments can be interpreted in two following aspects:

First, an answerer tends to answer questions in relevant categories in CQA services. Likewise, the chance of an answerer to answer questions in the irrelevant categories (i.e., without any relevance to his answered questions) is slim. For this reason, category-answerer indexes are constructed to filter irrelevant answerers. Significant positive effects of category-answerer indexes on filtering irrelevant

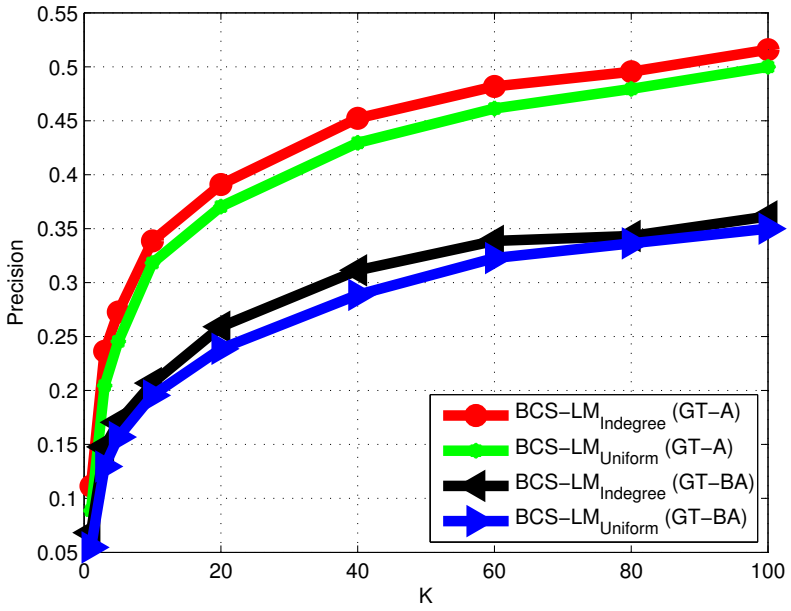

Figure 5.7: Prec@K of BCS-LM with different answerer priors.

answerers are found in the experiments, although only two category-answerer indexes are adopted.

Second, the expertise of an answerer in answering questions of a particular category probably works in answering questions of those relevant categories. In this connection, relevant categories are undertaken for expertise estimation, with TCS-LM being developed. The superiority of category-sensitive LMs, as demonstrated in the experiments, lies in more feasible computations and more accurate expertise location.

As discussed in the above, the incorporation of question category contributes to the efficiency of routing questions in CQA services. An implication is thus noted for refining the question category of CQA services. To give full play to question category, the structure of question category in CQA portals can be further explored to reflect relevance among various categories. In addition, askers might also be requested to choose a category as well as some similar categories for their questions.

5.5 Summary

In this chapter, we utilize question category to construct category-answerer indexes for filtering irrelevant answerers, and develop two category-sensitive LMs for estimating answerer expertise. The two attempts of category-answerer indexes and category-sensitive LMs enrich the framework of QR in CQA services by conceptualizing question category as an essential component.

Then, we conduct experiments on category-answerer indexes and category-sensitive LMs. Results reveal that: (1) category-answerer indexes produce a much shorter list of relevant answerers to be routed, with computational costs being substantially reduced; and (2) our category-sensitive LMs obtain more accuracies of expertise estimation, relative to QLLM and state-of-the-art baselines. The results of experiments prove that higher accuracies with lower costs are achieved due to the inclusion of question category in routing questions, which therefore provide empirical evidence to validate the incorporation of question category in QR for CQA services.

Looking forward, we plan to take into account the similarities among leaf categories and category hierarchies when constructing category-answerer indexes. Another potential fruitful area for future research might be to detect influence of question category on content quality in CQA services.

□ **End of chapter.**

Chapter 6

Question Structuralization

6.1 Motivation and Problem

Through routing, new questions obtain answers more efficiently. By accumulating more and more solved questions, CQA services retain a huge amount of human knowledge and user experience. Therefore, the organization and management of answered questions become a vital issue. Good organization not only facilitates users in browsing questions and acquiring knowledge, but also helps the system find similar questions effectively.

At present, most CQA portals organize questions in a list structure with category hierarchies [20, 70, 156] (e.g., Yahoo! Answers) or tags [43] (e.g., Quora). This "list-of-content" (list-based approach) is simple and straightforward, but ineffective for browsing and knowledge learning. Consider the following case: a user wants to spend his vacation in Edinburgh. He visits a CQA portal to explore which aspects are asked the most. In this scenario, he may browse some relevant categories like "Travel:United Kingdom:Edinburgh" to get useful information. He may also issue a query, such as "travel in Edinburgh", to search relevant questions. However, both browsing and searching give the user a list of relevant contents (e.g., questions shown in Table 6.1), not the direct knowledge. Thus, the user must read these contents, understand them, classify them into various topics, and obtain valuable knowledge himself. This is obviously inef-

Table 6.1: Sample questions about *Edinburgh*.

1. Where can i buy a **hamburger** in **Edinburgh**?
2. Where can I get a **shawarma** in **Edinburgh**?
3. How long does it take to drive between **Glasgow** and **Edinburgh**?
4. Whats the difference between **Glasgow** and **Edinburgh**?
5. Good **hotels** in **London** and **Edinburgh**?
6. Looking for nice , clean cheap **hotel** in **Edinburgh**?
7. Does anyone know of a reasonably cheap **hotel** in **Edinburgh** that is near to **Niddry Street South** ?
8. Who can recommend a affordable **hotel** in **Edinburgh City Center**?

fective and time-consuming.

The above problem calls for a new approach in structuralizing questions in CQA services, one which will facilitate users in seeking knowledge (e.g., travel information about Edinburgh) more effectively. Traditionally, we can utilize topic models [10] or social tagging [43] to structuralize questions. However, for topic models, it is not easy to control the granularity of topics [25], and it is difficult for users to interpret a topic based only on the multinomial distribution [98]. For social tagging, it is not applicable in many sites and has sparsity problem [132]. Thus, both topic models and social tagging are not suitable for structuralizing questions in CQA services.

In this chapter, we propose a novel hierarchical entity-based approach (i.e., "cluster entity tree" or CET) to structuralize questions in CQA services by leveraging an existing large-scale entity repository. Figure 6.1 shows how CET structuralizes questions in Table 6.1, where entities are shown in bold. In this CET, each node contains one (named) entity and a set of question IDs. With "edinburgh" as the root entity, Layer 1 includes all entities that co-occur with "edinburgh". Similarly, entities on Layer 2 co-occur with their parent entities on Layer 1 and the root entity "edinburgh". For exam-

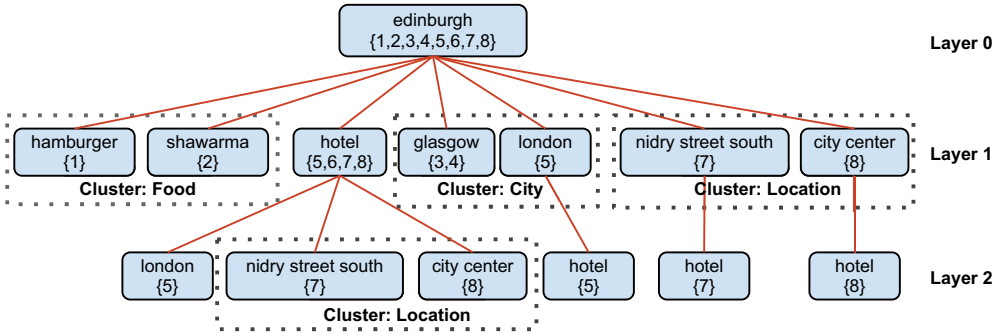

Figure 6.1: An CET constructed from questions about *Edinburgh*.

ple, "city center" co-occurs with "hotel" and "edinburgh" in Question 8. Deeper layers provide more specific topics about different aspects of Edinburgh (e.g., Edinburgh's hotels). As shown in Figure 6.1, the corresponding question IDs are also attached in each node. In addition, entities which share the same parent are clustered to different groups (see dashed rectangles in Figure 6.1)[1]. Through this hierarchical structure, we can easily browse corresponding questions and answers, and more effectively learn knowledge about Edinburgh, such as food and location. Moreover, since similar entities (and corresponding questions) are grouped on each layer, it also helps the system manage similar contents.

By utilizing a large-scale entity repository, CET avoids the granularity, interpretation, and sparsity problems. Entity repositories like Freebase[2] provide a large number of named entities across various pre-defined topics, which avoid the granularity and sparsity problems. In addition, they usually give descriptions of entities, which prevent interpretation problems from arising. Therefore, CET is more suitable for structuralizing questions in CQA services.

The work in [170], which automatically generates and updates topic terms to organize user generated content (UGC), is mostly related to our work. In that paper, given a root topic, subtopics and

[1]Single-node clusters are not surrounded by rectangles, like "hotel{5}" on Layer 2.

[2]http://www.freebase.com/

lower-level topics are extracted from UGC, which form a hierarchical structure in organizing corresponding UGC. However, more external sources are utilized to identify subtopics in [170]. In addition, relationships among subtopics under the same parent are not investigated. The metro maps proposed in [130] are also related to our work. Different from [130], we employ a large-scale entity repository to extract more meaningful and interpretable key terms (entities), which make each subtopic much easier to understand.

This chapter proceeds as follows. We detail our framework to construct CETs and show empirical results in Section 6.2. Section 6.3 and Section 6.4 evaluate the effectiveness of CET on knowledge organization from user and system aspects, respectively. We summarize the chapter in Section 6.5.

6.2 CET Construction

In this section, we formulate the framework to construct CET and show our empirical results. Firstly, we provide the definitions of the entity repository and CET.

Definition 5 (Entity Repository). *Let $ER = \{R, g\}$ be an entity repository, where R is a set of named entities and $g : R \times R \rightarrow [0, 1]$ is a mapping function that defines the similarity of any two entities.*

Note that we do not require a hierarchical structure in an ER (like Freebase), and only a similarity function is needed.

Definition 6 (Cluster Entity Tree). *Let D be a set of documents, $ER = \{R, g\}$ be an entity repository, e be an entity, a cluster entity tree $CET_e = (v_e, V, E, C)$ is defined as a tree structure, with the root node v_e, node set V, edge set E, and cluster set C. Each node $v_s \in V$ on CET_e includes an entity extracted from the set of documents $D_e \in D$ containing e, and a list $L(s)$ which stores the indexes of documents containing entity s and its superior entities. If v_s is*

v_t's parent node, entity t must co-occur with s and all of s' superior entities at least once in the same document. Each element of C (one cluster) includes a set of nodes which share the same parent node, and the entities within a cluster are more similar to each other than the entities in other clusters.

In our example in Section 6.1, D represents eight questions, v_e means the entity "edinburgh" and corresponding question IDs, V includes all nodes on the tree shown in Figure 6.1 except the root node, E contains all edges, and C represents the collection of dashed rectangles. In addition, $L(e)$ stores the indexes of all eight questions.

6.2.1 Framework

This section shows our three-step framework for constructing CET, which includes entity extraction, tree construction, and hierarchical entity clustering.

Entity extraction. We adopt an entity repository-based approach to extract entities, which addresses the "low-recall" problem for traditional methods (details will be given in Section 6.2.2). This approach involves two phases: candidate entity extraction and entropy-based filtering.

Candidate entity extraction. We employ the Stanford Parser[3] to parse each document to a parse tree. Then, we extract all noun phrases, pre-process them (including stemming), and extract the noun phrases which are included in our entity repository. In our experiments, we adopt a large-scale enterprise entity repository (i.e., NeedleSeek[4]).

Entropy-based filtering. The candidate entities generated from the last step may contain many false examples, which are not relevant to the main semantics of documents, like "we", "how do I", etc. To filter them, we propose an entropy-based method. Given a

[3]http://nlp.stanford.edu/software/lex-parser.shtml
[4]http://needleseek.msra.cn/

document with a category label (or tags, which are available in most UGC sites), we get the distributions of each candidate entity over all top categories. The entropy of a candidate entity e_i is calculated as follows:

$$Entropy(e_i) = -\sum_{c=1}^{|C|} P_c(e_i) \cdot \log(P_c(e_i)), \tag{6.1}$$

where $|C|$ is the number of top categories and $P_c(e_i)$ is the number of e_i in category c divided by all number of candidate entities in that category.

Top-ranked entities are general terms among categories. We set a threshold α and remove all candidate entities with entropy larger than α. The setting of α is a tradeoff: higher values will introduce more noise, while smaller values will lead to decreased recall. In our experiments, we empirically set α to 1.5 since it provides the most satisfying results in terms of both precision and recall.

Tree construction. Given an entity (e.g., "edinburgh"), we first search documents containing this entity and make the entity together with document IDs as the root node. Then, from searched documents we find all entities that co-occur with the root entity. These entities and corresponding document IDs form Layer-1 nodes of the entity tree (see the example in Figure 6.1). Afterwards, for each entity in Layer-1 nodes, we search entities that co-occur with it and its superiors, combine them with corresponding document IDs as new nodes, and put these new nodes under current nodes, which form Layer-2 nodes. Iteratively, we construct the entity tree with the given entity as the root.

Hierarchical entity clustering. Under the same parent, some entities[5] may share similar topics. Therefore, the final step is to hierarchically cluster entities with the same parents at different layers of entity trees. This step not only facilitates knowledge learning but also reduces the width of a tree. We follow the work in [54] and

[5]Here we use the entity to represent the node.

employ an agglomerative clustering algorithm for the following two reasons: (1) it is easy to implement as its time complexity is $O(N^2)$, in which N is the number of entities; and (2) there is no need to set the number of clusters. Any other advanced algorithms like spectral clustering [107] can also be applied, but that is not the emphasis of our approach.

For a set of entities with the same parent, we apply the agglomerative clustering algorithm[6] in Algorithm 3, in which three strategies are employed at line 4:

- AC-MAX: the similarity between entity e_i and entity e_j in one of the clusters (the first one) is larger than threshold θ_{max};
- AC-MIN: the similarity between entity e_i and any entity e_j in one of the clusters is larger than threshold θ_{min};
- AC-AVG: the mean similarity between entity e_i and any entity e_j in one of the clusters is larger than threshold θ_{avg}.

Algorithm 3 Agglomerative Clustering.

Input: A set of entities, and an entity repository which contains these entities.
Output: Clusters of input entities.

```
1:  Select one entity and create a new cluster containing the entity;
2:  while there are unclustered entities do
3:      Select the next entity e_i, create an empty candidate list, and then calculate
        the similarity between the entity and all existing clusters;
4:      if some condition is satisfied then
5:          Put the cluster index and corresponding similarity in the candidate list;
6:      end if
7:      if the candidate list is not empty then
8:          Put e_i in the cluster with the highest similarity;
9:      else
10:         Create a new cluster with e_i being the element.
11:     end if
12: end while
```

[6] We modify the clustering algorithm in [54] slightly to assign a unique cluster for each entity.

In our entity repository, the similarity between two entities is computed using the approach in [133], which estimates the similarity of two terms according to their *first-order* and *second-order* co-occurrences. For example, "such as NP, NP" is a good pattern for detecting similar entities using *first-order* co-occurrences. In addition, if two entities usually co-occur with a third entity (*second-order* co-occurrence), these two entities are likely to be similar. To construct similarity functions, pattern-based approaches [110, 159] utilize *first-order* co-occurrences while distributional similarity approaches [114, 116] employ *second-order* co-occurrences. In the following, we briefly introduce the pattern-based approach (PB) and the distributional similarity approach (DS) in [133].

PB. Some well-designed patterns are leveraged to extract similar entities from a huge repository of webpages. The set of terms extracted by applying a pattern once is called a raw semantic class (RASC). Given two entities t_a and t_b, PB calculates their similarity based on the number of RASCs containing both of them [159]:

$$sim(t_a, t_b) = \log(1 + \sum_{i=1}^{r_{ab}} P_{ab_i})) \cdot \sqrt{idf(t_a) \cdot idf(t_b)}, \qquad (6.2)$$

where $idf(t_a) = \log(1 + \frac{N}{C(t_a)})$, P_{ab_i} is a pattern which can generate RASC(s) containing both term t_a and term t_b, r_{ab} is the total number of such patterns, N is the total number of RASCs, and $C(t_a)$ is the number of RASCs containing t_a. The above similarity is normalized using the following function:

$$sim_{PB}(t_a, t_b) = \frac{\log(sim(t_a, t_b))}{2\log(sim(t_a, t_a))} + \frac{\log(sim(t_a, t_b))}{2\log(sim(t_b, t_b))}. \qquad (6.3)$$

DS. The DS approach assumes that terms appearing in similar contexts tend to be similar. In this approach, a term is represented by a feature vector, with each feature corresponding to a context in which the term appears. The similarity between two terms is computed as the similarity between their corresponding feature vectors.

Table 6.2: Category mapping between Yahoo! Answers and FreeBase.

Yahoo! Answers	FreeBase
Cars & Transportation	Aviation, Transportation, Boats Spaceflight, Automotive, Bicycles, Rail
Computers & Internet	Computer, Internet
Sports	Soccer, Olympics, Sports, American football, Baseball, Basketball, Ice Hockey, Martial Arts, Cricket, Tennis, Boxing, Skiing
Travel	Travel, Location, Transportation

Jaccard similarity is employed to estimate the similarity between two terms. Suppose the feature vector of t_a and t_b are **x** and **y**, respectively, then we have:

$$sim_{DS}(t_a, t_b) = \frac{\sum_i \min(x_i, y_i)}{\sum_i x_i + \sum_i y_i - \sum_i \min(x_i, y_i)}. \quad (6.4)$$

Shi et al. [133] found that PB performed better when dealing with proper nouns, while DS was relatively good at estimating similarity of other types of entities. The similarity function in our ER follows the suggestion of [133]: if at least one entity is proper noun, PB is employed; otherwise DS is used.

6.2.2 Experiments

Setup. We evaluate the performance of our framework by employing questions from Yahoo! Answers. We collect 54.7 million questions from all 26 top categories in Yahoo! Answers, which consist of question titles and corresponding categories. From these questions, we construct the following two test sets for evaluating entity extraction and entity clustering:

Set EE. This set is employed to evaluate the performance of entity extraction. It contains 520 randomly sampled questions, 20 from

Table 6.3: Number of questions and entities in Set EC.

Category	Number of Questions	Number of Entities
Cars & Transportation	1,220,427	3,267,596
Computers & Internet	2,912,280	7,324,655
Sports	2,363,758	6,230,868
Travel	1,347,801	3,728,286

each top category. We manually label entities for each question.

Set EC. This set is constructed to automatically evaluate hierarchical entity clustering and select the best clustering strategy. The construction process is as follows. First, we map the four categories of Yahoo! Answers with some categories of Freebase manually, as shown in Table 6.2. Second, from questions at each top category of Yahoo! Answers, we extract entities which appear exactly once in the corresponding Freebase categories. For instance, if an entity is extracted from questions in the category *Computers & Internet*, and it appears two or more times in *Computer* and *Internet* categories in Freebase, it will be filtered. Therefore, each entity is attached with a unique Freebase category label (i.e., the ground truth for clustering). Questions containing at least two entities are selected for Set EC. Table 6.3 reports the statistics. Intuitively, entities with a same Freebase category label should be in one cluster.

Note that Set EC covers only a small set of real entities, and clustering on Set EC is partial clustering. However, it leverages Freebase labels and avoids manual labeling, which is time-consuming. Furthermore, partial clustering results are enough for evaluating different strategies' performance and choosing the best strategy.

Following the common practice, we evaluate entity extraction using precision, recall, and F_1 scores. For evaluating entity clustering, we adopt B-cubed metrics. As reported in [5], B-cubed metrics are more suitable than traditional metrics, such as NMI and purity.

Table 6.4: Entity extraction for various methods.

Method	Precision	Recall	F_1
Stanford NER	0.750	0.155	0.257
FIGER	0.763	0.154	0.256
Freebase	0.644	0.595	0.619
Ours	0.647	0.809	0.719

Results of entity extraction. Two ERs (i.e., ours and Freebase) are employed in entity extraction for comparison. In addition, we compare our approach to Stanford named entity recognizer (NER) [36] and the fine-grained entity recognition (FIGER) algorithm proposed by Ling et al. [77]. Table 6.4 reports the results of different methods. We can find that Stanford NER and FIGER get a relatively high precision in extracting entities. However, their recalls are very low, and only about 15% of entities are recognized. With the help of entity repositories, recall is significantly improved with a small decrease of precision. Therefore, the F_1 scores of entity-based approaches are much higher. This observation highlights the significant advantage of utilizing entity repositories in entity extraction and the effectiveness of our approach. As our ER performs better than Freebase, we adopt it as our entity repository in the following evaluations.

Results of hierarchical entity clustering. Table 6.5 and Table 6.6 report the count of clusters, B-Cubed Precision, Recall, and F_1 for different layers of clustering across four categories using AC-MAX. In our experiments, AC-MAX performed better than AC-MIN and AC-AVG. Therefore, we only report the results of AC-MAX here. For AC-MAX, we changed the settings of θ from 0.01 to 0.9, and the best performance was achieved when θ_{max} was set to around 0.1. We find that although AC-MAX's accuracy varies across categories (e.g., the F_1 of *Transportation* is much higher than

Table 6.5: Clustering results of *Cars & Transportation* and *Computer & Internet* using AC-MAX (θ_{max}=0.1).

Level	*Cars & Transportation*				*Computer & Internet*			
	Count	**P**	**R**	**F_1**	**Count**	**P**	**R**	**F_1**
1	1,281	0.948	0.868	0.897	3,064	0.913	0.664	0.743
2	1,202	0.989	0.956	0.965	11,344	0.961	0.842	0.879
3	858	1.000	0.981	0.988	8,184	0.978	0.899	0.920
4	1,776	1.000	0.980	0.986	3,648	0.990	0.908	0.934
5	NA	NA	NA	NA	2520	1.000	0.952	0.968
Total	5,117	0.984	0.946	0.959	28,760	0.968	0.857	0.891

that of *Travel*), it performs well overall. Thus, we adopt AC-MAX with $\theta = 0.1$ for hierarchical entity clustering.

6.3 User Study

In this section, we investigate the influence of CET on content browsing and knowledge learning from a user study. In the study, we design 24 tasks in four popular Yahoo! Answers categories (see Table 6.7). For each category, we design three knowledge-learning tasks and three question-search tasks. A knowledge-learning task asks for some knowledge about a main entity from question texts. For instance, "find the games running on macbook pro" requires game names as the answer, where the main entity is "macbook pro". A question-search task, however, asks users to find similar questions to the question in the task. For example, "questions about who will win the MVP in the NBA this year" asks for finding similar questions, and filling their question IDs as the answer. For each task, we give some suggested keywords (entities) to facilitate information gathering.

To evaluate user experience, we ask participants to each fill out a

Table 6.6: Clustering results of *Sports* and *Travel* using AC-MAX (θ_{max}=0.1).

Level	*Sports*				*Travel*			
	Count	**P**	**R**	$\mathbf{F_1}$	**Count**	**P**	**R**	$\mathbf{F_1}$
1	890	0.941	0.883	0.901	748	0.972	0.653	0.743
2	636	0.978	0.964	0.963	200	0.974	0.730	0.798
3	492	0.965	0.882	0.899	120	1.000	0.833	0.890
4	1,080	0.978	0.844	0.881	NA	NA	NA	NA
5	NA	NA	NA	NA	NA	NA	NA	NA
Total	3,098	0.965	0.886	0.907	1,068	0.976	0.688	0.770

questionnaire after each task. Following the work in [61], we collect information based on five aspects: familiarity, easiness, satisfaction, adequate time, and helpfulness. A 5-point Likert scale is designed for each questionnaire. A "5" means the participant totally agrees while a "1" means the participant totally disagrees.

6.3.1 Setup

Programs. We develop two programs in our user study. One is CET-based (see Figure 6.2), and the other is traditional list-based.

The CET-based program works as follows. When the user types an entity name and clicks the search button, the corresponding CET with the searched entity as the root will be displayed. In each CET, we label each entity cluster with the top three most frequent entities. When the user clicks on one single-node cluster, the corresponding questions which contain the entity and its superior entities will be displayed to the right of CET. In every task, we allow participants 3 minutes to find information and fill in the answers in the answer panel. When a participant collects all of the information she wants, she may click the submission button to enter the questionnaire part. If the participant fails to submit her answer within 3 minutes, the software will store the current answer, disable the submission but-

ton, and forcibly jump to the questionnaire part. After the participant finishes the questionnaire, the program provides the next task. When all tasks are finished, the program exits automatically. The list-based program follows the same process.

The interface of list-based program is similar, but the CET display area is replaced by a flat-ranked list. In addition, the list-based program searches questions by utilizing Apache Lucene[7]. The standard analyzer and the default search algorithm are adopted. For each query, top 200 most relevant questions are retrieved.

Data. We extract 70,195 questions which contain at least one of the 24 main entities (see Table 6.7) in the four categories. For each question, we extract the entities with the help of our entity repository. For each main entity, we build the corresponding CET from all extracted questions.

Participants. Sixteen volunteers are invited to participate in the user study. These participants are all well educated with either having a Bachelor's or Master's degrees or currently studying in their PhD programs. Moreover, their background covers a wide range of disciplines from natural sciences to social sciences, as well as engineering. All volunteers are first briefly informed of the research design and taught how to use two programs. To familiarize the participants with our programs promptly, we provide demonstrations using sample entities. Each volunteer is asked to finish 12 tasks (6 knowledge-learning tasks and 6 question-search tasks) using the CET-based program and 12 other tasks using the list-based program in random order. Thus, each task is finished by exactly 8 different participants using each program.

[7] http://lucene.apache.org/core/

Table 6.7: User study tasks (E: knowledge-learning tasks; S: question-search tasks).

ID	Task Title	Category	Main Entity	Type
1	Find the names of games running on macbook pro	Computer & Internet	Macbook Pro	E
2	Find which components of thinkpad notebooks are usually asked	Computer & Internet	Thinkpad	E
3	How to ps body using photoshop cs2	Computer & Internet	Photoshop CS2	E
4	Questions about the best canon laser printer for a mac	Computer & Internet	Laser printer	S
5	Questions about how to connect Xbox 360 to Laptop or PC using a router	Computer & Internet	Xbox 360	S
6	Questions about green screen problem of Windows Movie Maker	Computer & Internet	Windows Movie Maker	S
7	Find the cities compared with Edinburgh	Travel	Edinburgh	E
8	Find the names of animals on myrtle beach	Travel	Myrtle Beach	E
9	Find the names of cities in Portugal	Travel	Portugal	E
10	Questions about looking for good hostels in Madrid	Travel	Madrid	S
11	Questions about how to get a low price ticket to Hong Kong Disneyland	Travel	Disneyland	S
12	Questions about how to go to Chinatown in Chicago	Travel	Chicago	S
13	Find the brand of running shoes that users have asked	Sports	Running shoes	E
14	Find football players that compared with messi	Sports	Messi	E
15	Find the names of skiing places that users have asked	Sports	Skiing	E
16	Questions asking horse racing website	Sports	Horse racing	S
17	Questions about who will win the MVP in NBA this year	Sports	NBA	S
18	Questions about when is the next match between Barcelona and Real Madrid	Sports	Real Madrid	S
19	Find the brand of cars that have been compared with Toyota	Cars & Transportation	Toyota	E
20	Which aspects of Jeep Wrangler have been asked	Cars & Transportation	Jeep Wrangler	E
21	Finding the names of sports cars being asked	Cars & Transportation	Sports cars	E
22	Questions which compare Mercedes Benz and BMW	Cars & Transportation	Mercedes Benz	S
23	Questions about the price to tow a suv from Newark to Florida	Cars & Transportation	SUV	S
24	Questions about How to reset the oil light for a 95 Honda civic	Cars & Transportation	Honda Civic	S

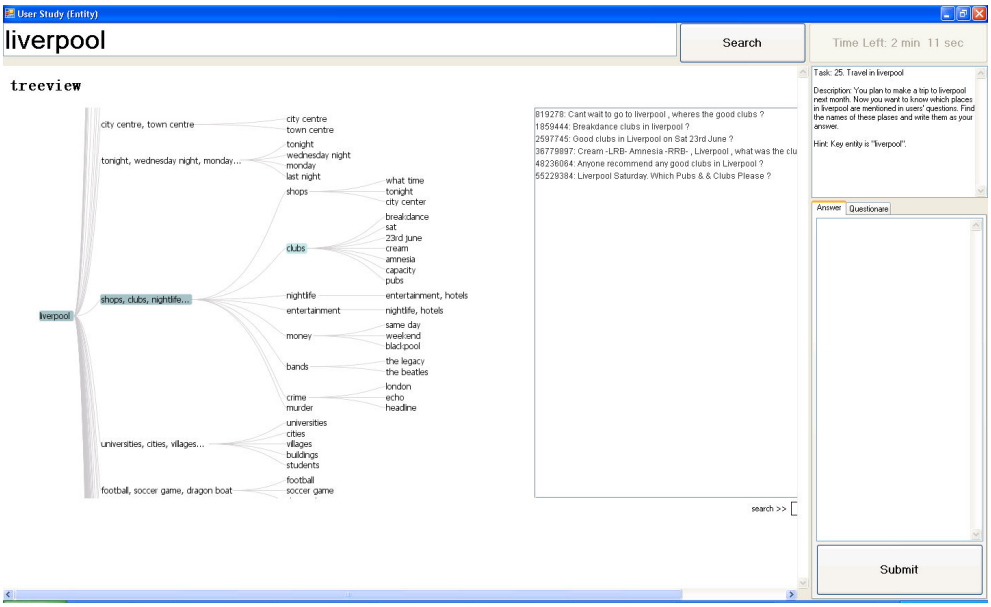

Figure 6.2: The interface of CET-based program.

6.3.2 Results and Discussions

Table 6.8 reports the user study results, where we give the statistics for users' performance with the two programs. We evaluate from the number of queries issued, number of answers found, the precision of answers, and query time for each task. As our 24 tasks contain both knowledge-learning tasks and question-search tasks, we report their results separately. Z-tests are employed for significance tests.

From Table 6.8, we observe that more queries are issued in the knowledge-learning tasks than in the question-search tasks using both programs. However, the CET-based program reduces the number of queries substantially in both tasks. Because the CET-based program provides a series of clustered entities, it helps users further refine queries by clicking on entities rather than reconstructing new queries. However, the list-based program only lists relevant questions, and users must issue new queries according to returned questions.

By using the CET-based program, volunteers find more answers

Table 6.8: User study results.

	Knowledge-learning Tasks		Question-search Tasks	
	CET-based	**List-based**	**CET-based**	**List-based**
# Queries	2.99	4.47	2.56	3.38
# Answers	8.32	6.06	10.60	10.92
Precision	0.38	0.19	0.40	0.44
Time	136.44	121.87	103.71	87.75

in knowledge-learning tasks ($z = 1.69, p < 0.05$). The reason is that the CET-based program clusters similar results in the same group, and if the user finds one answer, she can easily obtain more answers. On the contrary, the list-based program returns a list of questions, and users must find answers question-by-question. For question-search tasks, users of the list-based program find more answers, but the difference is not significant ($z = 0.19$). As the list-based program returns similar questions as top-ranked results, users are able to fill in answers easily. For CET-based program users, they have to find corresponding key entities in the CETs first. Therefore, they spend more time (the fourth row in Table 6.8) finding entities and less time filling answers. It is worth noting that our GUI prototype for CET is non-optimal, and users' searching time on CET-based program can be further reduced with a better user interface.

The precision of answers from CET-based program users is twice of that from list-based program users ($z = 4.15, p < 0.0001$) in knowledge-learning tasks, which demonstrates the advantage of CET in helping knowledge-learning. For question-search tasks, CET-based program users perform slightly worse than list-based program users, but the difference is not significant ($z = 0.48$). Since users of the CET-based program spend more time finding entities, they have limited time to check the answers.

In both tasks, users spend more time on the CET-based program.

Table 6.9: Questionnaire results.

	Knowledge-learning Tasks		Question-search Tasks	
	CET-based	**List-based**	**CET-based**	**List-based**
Familiarity	3.18	3.22	3.07	3.28
Easiness	3.64	3.66	4.10	4.06
Satisfaction	3.70	2.94	3.86	3.44
Enough Time	3.87	3.83	4.44	4.54
Helpfulness	4.16	3.03	4.31	3.71

According to users' post-user-study feedbacks, a few volunteers reported that they sometimes spent a considerable amount of time on finding entities from CETs. However, one positive observation is that most users find "the entity-based interface" very interesting, which stimulates them to spend more time on exploring answers.

The questionnaires reveal more about user experience on these two programs (see Table 6.9). Users' responses to task familiarity and easiness are similar. However, users of entity-based interface are more satisfied in both knowledge-learning tasks ($z = 3.98, p < 0.0001$) and question-search tasks ($z = 1.38$), and they feel that entity-based interface is more helpful in finding answers for both knowledge-learning tasks ($z = 6.47, p < 0.0001$) and question-search tasks ($z = 2.55, p < 0.01$). These promising observations show that CET helps knowledge learning greatly through structuralizing content.

6.4 CET-based Question Re-ranking

In this section, we show that CET also helps systems to better retrieve information [135, 154, 136] through re-ranking. In the following, we continue to use Yahoo! Answers as a test case.

Intuitively, questions sharing similar topics should be ranked sim-

ilarly. However, traditional question retrieval models [19] such as the query likelihood language model (QLLM) and the vector space model (VSM) do not capture key semantics nor give more weight to entity terms. CET provides a feasible way to address this issue. By utilizing CET, entities are given more weight while trivial words are not. In addition, through clustering questions with similar topics, those questions which are ranked lower will be elevated by their top-ranked neighbors.

Algorithm 4 illustrates the re-ranking algorithm in detail. We first extract entities from each question, and construct a entity co-occurrence graph (Line 1). Then, we calculate the PageRank score of each entity (Line 2). Line 3-5 check whether the query q contains at least one entity. If the answer is no, we return to the original ranking. Otherwise, we identify the key entity in q (Line 6) and construct the CET cet_e whose root entity is e (Line 7). Line 8-16 iteratively put questions in corresponding clusters of cet_e. In Line 8, we first build an entity chain for question q_i, in which entities of q_i are ranked according to their PageRank scores. Afterwards, the first entity $\hat{e}$, which is not similar to e (the similarity is calculated in Section 6.2.1, and the threshold of similarity is set to 0.1), is picked up as the main aspect of e, and q_i is grouped into the corresponding cluster on cet_e (Lines 7-8). If $\hat{e}$ does not exist, we put q_i in a new cluster (Lines 13-14). Then, we rank all clusters according to their first elements' original rankings (Line 18), and output the final re-ranked list (Line 19).

We perform our re-ranking on 160 randomly selected questions from *Computers & Internet* and *Travel* categories of our data set[8]. Each category contains 80 questions. All other questions in these two categories constitute the question collection Q. For each question, we first employ VSM and QLLM, respectively, to retrieve the top 15 results and then obtained manual judgments. Given a retrieved question by VSM or QLLM, two assessors are asked to label

[8]These 160 questions are not used for constructing the entity co-occurrence graph.

Algorithm 4 CET-based search results re-ranking.

Input: Query q, question collection Q, a ranked list of k relevant questions $Q_q = \{q_1, q_2, ..., q_k\}$ to q, an entity repository ER, an empty list Θ.

Output: A new ranked list of questions.

Extract entities from each question of Q and construct an entity co-occurrence graph;
Get the PageRank score of each entity;
if there is no entity e in q **then**
    return Q_q;
else
    Identify the key entity e from q which has the highest PageRank score;
    Construct the CET cet_e from Q based on ER;
    for each question q_i **do**
        For all entities in q_i, build an entity chain C in descending order of PageRank scores;
        From C extract the first entity $\hat{e}$ that is not similar to e;
        if $\hat{e}$ exists **then**
            Put q_i in the corresponding cluster of nodes;
        else
            Put q_i in Θ;
        end if
    end for
end if
Rank all clusters on cet_e according to their first elements' original ranking;
Output the final ranking cluster by cluster and append the questions in Θ at the last.

Table 6.10: Re-ranking results for VSM and QLLM (* means that $p < 0.05$ in students' t-test).

	VSM	Re-ranking	QLLM	Re-ranking
MRR	0.3838	**0.4195*** (9.30%)	0.3593	**0.3889*** (8.24%)
MAP	0.3376	**0.3558*** (5.39%)	0.3326	**0.3479*** (4.60%)
Prec@1	0.2500	**0.3125*** (25.00%)	0.2438	**0.2688*** (10.25%)

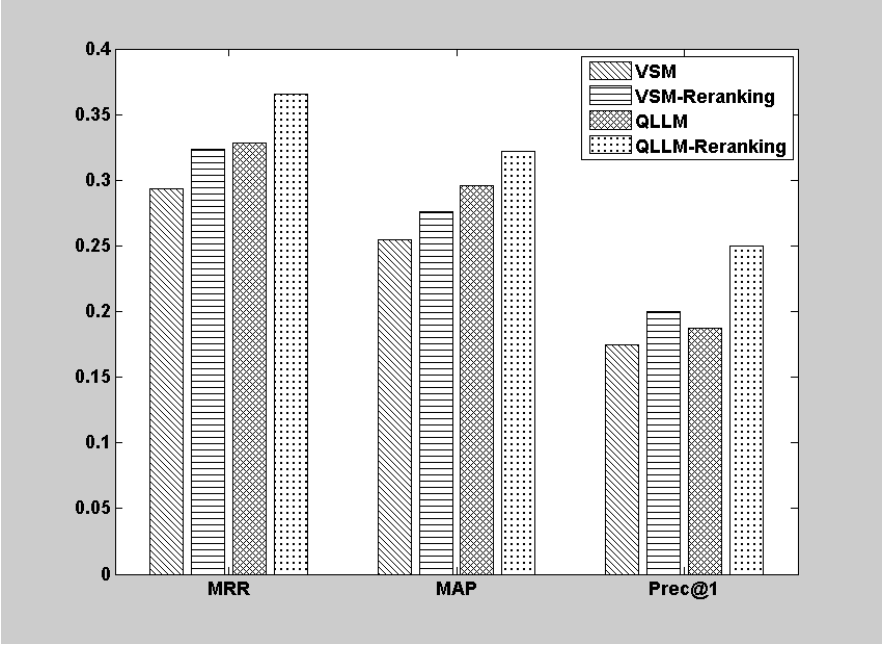

Figure 6.3: Re-ranking results of *Computer & Internet*.

it "relevant" or "irrelevant". If their annotations are contradictory, the third assessor is involved to determine the final label.

We re-rank these questions using Algorithm 4. Table 6.10 shows the results of MRR, MAP, and Prec@1. We can see that CET-based re-ranking improves the performance of standard retrieval models substantially. For VSM, our re-ranking boosts MRR and MAP by 9.3% and 5.4%, respectively. It is worth noting that our re-ranking improves Prec@1 significantly: from 0.25 to 0.31. The reason is that traditional methods may give relatively low weights to the key terms (entities), while CET-based re-ranking addresses the problem. QLLM and re-ranking report similar results.

Figure 6.3 and Figure 6.4 illustrate the performance of various approaches across categories. We find that our re-ranking is neither category-biased nor algorithm-biased, yet it performs better than original models on both categories. The above results demonstrate that, by utilizing the hierarchical entity-based approach, CET greatly

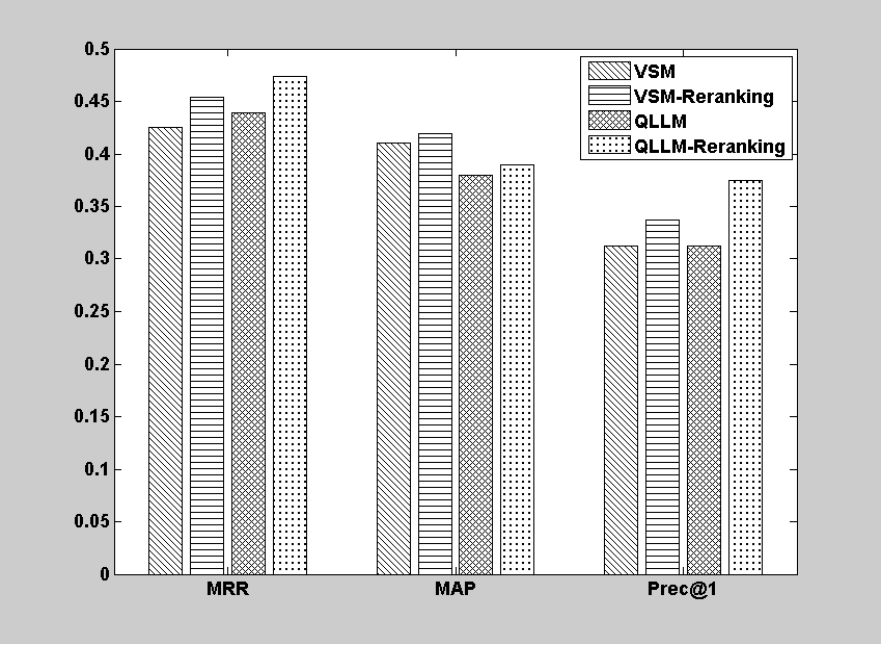

Figure 6.4: Re-ranking results of *Travel*

improves the retrieval performance of these two standard models.

6.5 Summary

Traditional list-based organization of questions in CQA services is not effective for content browsing and knowledge learning due to large volume of documents. To address this problem, we propose a novel hierarchical entity-based approach to structuralize questions. By using a large-scale entity repository, we construct a three-step framework to organize knowledge in "cluster entity trees". Experimental results show the effectiveness of the framework in constructing CET. We further evaluate the performance of CET on knowledge organization from both user and system aspects. Our user study demonstrates that, with CET-based organization, users perform significantly better in knowledge learning than by using the list-based approach. In addition, CET boosts systems' question search perfor-

mance substantially through re-ranking.

To the best of our knowledge, this work is the first attempt to utilize entities to structuralize questions in CQA services. However, there are some limitations to be improved in the future work. First, we employ Yahoo! Answers as our test data, in which questions (documents) are usually short. We observe that nearly 92% of all 54.7 million questions contains one to four entities, which means the depths of CETs are usually not so deep. However, long questions will lead to deep CETs and hinder users' knowledge learning. Second, our current entity extraction focuses on named entities instead of canonical entities. In the future, we plan to employ document summarization techniques to shorten the depths of CETs. We also aim to incorporate semantic analysis and normalize named entities to canonical entities.

□ **End of chapter.**

Chapter 7

Conclusion

7.1 Summary

In this thesis, we propose a computational framework to process questions in community question answering (CQA) services. This framework, which aims to help users obtain information and learn knowledge more effectively, consists of three components: question popularity analysis and prediction, question routing, and question structuralization. The first component analyzes the factors affecting question popularity (QP) and predicts QP when new questions are asked. We then utilize question routing (QR), the second component of the framework, to route questions to appropriate answerers for efficient answering. The third component proposes a hierarchical entity based approach to structuralize questions into cluster entity trees (CETs) to help both users and the system access information more effectively.

The first component involves two studies. In Study 1, we analyze the factors influencing QP and find that the interaction of users and topics leads to the difference of QP. Based on the findings of Study 1, Study 2 proposes a mutual reinforcement-based label propagation algorithm to predict QP using features from question texts and asker profiles. Empirical results demonstrate that our algorithm is more effective in distinguishing high-popularity questions from low-popularity ones than other state-of-the-art algorithms, such as

the stochastic gradient boosted tree and the harmonic function.

The second component proposes a framework to route new questions to potential answerers in CQA services. The proposed QR framework considers both answerer expertise on routed questions and answerer availability for providing answers in a given range of time. To estimate answerer expertise, we propose three models. The first one is derived from the query likelihood language model, and the latter two models utilize answer quality to refine the first model. To estimate answerer availability, we employ an autoregressive model. Experimental results show that it is effective to route questions to answerers who have answered similar questions previously. It also demonstrates that leveraging answer quality can greatly improve the performance of QR. In addition, utilizing similar answerers' answer qualities on similar questions provides more accurate expertise estimation and thus obtains better QR performance. Moreover, answerer availability estimation further boosts the performance of QR.

User expertise estimation is essential to QR. However, current approaches employ full profiles to estimate all answerers' expertise, which is ineffective and time-consuming. To address this problem, we utilize question categories to construct category-answerer indexes for filtering irrelevant answerers and develop category-sensitive language models for estimating user expertise. We conduct experiments on large scale data sets and the results reveal that: (1) category-answerer indexes produce a much shorter list of relevant answerers to be routed, with computational costs substantially reduced; and (2) category-sensitive language models obtain more accurate expertise estimations, relative to query likelihood language model and state-of-the-art algorithms, including CBLM and LDALM. The results of experiments prove that higher accuracies with lower costs are achieved due to the inclusion of question categories in routing questions.

In the third component, we propose a novel hierarchical entity-

based approach to structuralize questions in CQA services. Traditional list-based organization of questions is not effective for content browsing and knowledge learning due to the large volume of documents. To address this problem, we utilize a large-scale entity repository, and construct a three-step framework to structuralize questions in CETs. Experimental results show the effectiveness of the framework in constructing CET. We further evaluate the performance of CET on knowledge organization from both user and system aspects. From a user aspect, our user study demonstrates that, with CET-based organization, users perform significantly better in knowledge learning than by using the list-based approach. From a system aspect, CET substantially boosts the performance on question search through re-ranking.

7.2 Future Work

Although a substantial number of promising achievements on techniques and applications have been presented in this thesis, there are several research directions we can follow to further improve the question processing framework in the future.

Firstly, we aim to explore more salient features of QP as current features are still not sufficient to achieve satisfying performance. In addition, semantics can be incorporated to improve the proposed algorithm for predicting QP. We also plan to utilize QP to improve question search and dialog analysis in CQA services.

Secondly, further study is needed to investigate the performance of our QR framework across different CQA portals like Baidu Zhidao and Quora, and different question domains (instead of the *Computer & Internet* category). In addition, more advanced availability estimation model can be utilized to make the prediction more accurate. Moreover, we plan to construct category-answerer indexes taking account of similarities among leaf categories and category hierarchies.

Thirdly, we attempt to employ document summarization techniques to handle long questions and shorten the depth of CET, and normalize named entities to canonical entities. In addition, we plan to utilize CET for better question summarization, question visualization, and answer exploration. Meanwhile, we also aim to incorporate semantic analysis in CET construction and link CET to large-scale knowledge base (e.g., Google Knowledge Graph[1]) for practical use.

Fourthly, users in different communities (e.g., categories in CQA services) may own different characteristics and behaviors, which may have various influences to question processing. In the future, we plan to apply our question processing framework to more communities, and explore the results of QP prediction, question routing, and question structuralization from different communities.

A trend toward incorporate social network in CQA services has been raised during recent years, like Facebook Questions and Quora. This interaction brings more research issues for question processing, e.g., how to utilize social relationships in question retrieval and quality prediction. The influence of social network on question routing is also worth further investigation.

□ **End of chapter.**

[1] http://www.google.com/insidesearch/features/search/knowledge.html

Bibliography

[1] L. A. Adamic, J. Zhang, E. Bakshy, and M. S. Ackerman. Knowledge sharing and Yahoo! Answers: everyone knows something. In *Proceedings of the 17th International Conference on World Wide Web*, pages 665–674, 2008.

[2] G. Adomavicius and A. Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6):734–749, 2005.

[3] E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 183–194, 2008.

[4] E. Agichtein, Y. Liu, and J. Bian. Modeling information-seeker satisfaction in community question answering. *ACM Transactions on Knowledge Discovery from Data*, 3(2):1–27, 2009.

[5] E. Amigó, J. Gonzalo, J. Artiles, and F. Verdejo. A comparison of extrinsic clustering evaluation metrics based on formal constraints. *Information Retrieval*, 12(4):461–486, 2009.

[6] K. Balog, L. Azzopardi, and M. de Rijke. Formal models for expert finding in enterprise corpora. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Re-*

search and development in Information Retrieval, pages 43–50, 2006.

[7] J. Bian, Y. Liu, E. Agichtein, and H. Zha. A few bad votes too many?: Towards robust ranking in social media. In *Proceedings of the 4th International Workshop on Adversarial Information Retrieval on the Web*, pages 53–60, 2008.

[8] J. Bian, Y. Liu, E. Agichtein, and H. Zha. Finding the right facts in the crowd: factoid question answering over social media. In *Proceedings of the 17th International Conference on World Wide Web*, pages 467–476, 2008.

[9] J. Bian, Y. Liu, D. Zhou, E. Agichtein, and H. Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In *Proceedings of the 18th International Conference on World Wide Web*, pages 51–60, 2009.

[10] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent Dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[11] M. J. Blooma, A. Y. K. Chua, and D. H.-L. Goh. A predictive framework for retrieving the best answer. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, pages 1107–1111, 2008.

[12] M. J. Blooma, A. Y.-K. Chua, D. H.-L. Goh, and Z. Ling. Towards a hierarchical framework for predicting the best answer in a question answering system. In *Proceedings of the 10th International Conference on Asian Digital Libraries*, pages 497–498, 2007.

[13] A. Blum and T. Mitchell. Combining labeled and unlabeled data with co-training. In *Proceedings of the Eleventh Annual*

Conference on Computational Learning Theory, pages 92–100, 1998.

[14] M. Bouguessa, B. Dumoulin, and S. Wang. Identifying authoritative actors in question-answering forums: the case of Yahoo! Answers. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 866–874, 2008.

[15] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. In *Proceedings of the Seventh International Conference on World Wide Web*, pages 107–117, 1998.

[16] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, 2002.

[17] S. Budalakoti. *Authority identification in online communities and social networks*. PhD thesis, The University of Texas at Austin, 2013.

[18] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, 2002.

[19] X. Cao, G. Cong, B. Cui, and C. S. Jensen. A generalized framework of exploring category information for question retrieval in community question answer archives. In *Proceedings of the 19th International Conference on World Wide Web*, pages 201–210, 2010.

[20] X. Cao, G. Cong, B. Cui, C. S. Jensen, and Q. Yuan. Approaches to exploring category information for question retrieval in community question-answer archives. *ACM Transactions on Information Systems*, 30(2):7:1–7:38, 2012.

[21] X. Cao, G. Cong, B. Cui, C. S. Jensen, and C. Zhang. The use of categorization information in language models for question retrieval. In *Proceeding of the 18th ACM Conference on Information and Knowledge Management*, pages 265–274, 2009.

[22] V. Cardellini, M. Colajanni, and P. S. Yu. Dynamic load balancing on web-server systems. *IEEE Internet Computing*, 3(3):28–39, 1999.

[23] W. Chan, W. Yang, J. Tang, J. Du, X. Zhou, and W. Wang. Community question topic categorization via hierarchical kernelized classification. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 959–968, 2013.

[24] W. Chan, X. Zhou, W. Wang, and T.-S. Chua. Community answer summarization for multi-sentence question with group l_1 regularization. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 582–591, 2012.

[25] M. Chen, X. Jin, and D. Shen. Short text classification improved by learning multi-granularity topics. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 1776–1781, 2011.

[26] A. Y. Chua and S. Banerjee. So fast so good: An analysis of answer quality and answer speed in community question-answering sites. *Journal of the American Society for Information Science and Technology*, 64(10):2058–2068, 2013.

[27] H. Chung, Y.-I. Song, K.-S. Han, D.-S. Yoon, J.-Y. Lee, H.-C. Rim, and S.-H. Kim. A practical QA system in restricted domains. In *Proceedings of the ACL 2004 Workshop on Question Answering in Restricted Domain*, pages 39–45, 2004.

[28] D. Correa and A. Sureka. Fit or unfit: analysis and prediction of 'closed questions' on Stack Overflow. In *Proceedings of the first ACM Conference on Online Social Networks*, pages 201–212, 2013.

[29] G. Dror, Y. Koren, Y. Maarek, and I. Szpektor. I want to answer; who has a question?: Yahoo! Answers recommender system. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1109–1117, 2011.

[30] G. Dror, Y. Maarek, and I. Szpektor. Will my question be answered? predicting "question answerability" in community question-answering sites. In *Machine Learning and Knowledge Discovery in Databases*, volume 8190 of *Lecture Notes in Computer Science*, pages 499–514. Springer, 2013.

[31] Q. Du, Q. Wang, J. Cheng, Y. Cai, T. Wang, and H. Min. Explore social question and answer system based on relationships in social network. In *Proceedings of the Fourth International Conference on Emerging Intelligent Data and Web Technologies*, pages 490–495, 2013.

[32] H. Duan, Y. Cao, C.-Y. Lin, and Y. Yu. Searching questions by identifying question topic and question focus. In *Proceedings of 46th Annual Meeting of the Association for Computational Linguistics: Human Language Tchnologies (ACL:HLT)*, pages 156–164, 2008.

[33] D. Eck, P. Lamere, T. Bertin-Mahieux, and S. Green. Automatic generation of social tags for music recommendation. In *Advances in Neural Information Processing Systems*, pages 385–392, 2007.

[34] C. Fellbaum. *WordNet: An Electronic Lexical Database*. Language, Speech and Communication. MIT Press, 1998.

[35] P. Fichman. *Information Quality on Yahoo! Answers*, chapter 18, pages 295–307. Approaches and Processes for Managing the Economics of Information Systems. Idea Group Publishing, 2014.

[36] J. R. Finkel, T. Grenager, and C. Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 363–370, 2005.

[37] J. H. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, 29(5):1189–1232, 2001.

[38] J. H. Friedman. Stochastic gradient boosting. *Computational Statistics & Data Analysis*, 38(4):367–378, 2002.

[39] R. Gazan. Specialists and synthesists in a question answering community. *Proceedings of the American Society for Information Science and Technology*, 43(1):1–10, 2006.

[40] R. Gazan. Microcollaborations in a social Q&A community. *Information Processing and Management*, 46(6):693–702, 2010.

[41] J. Guo, S. Xu, S. Bao, and Y. Yu. Tapping on the potential of Q&A community by recommending answer providers. In *Proceeding of the 17th ACM Conference on Information and Knowledge Management*, pages 921–930, 2008.

[42] L. Guo and X. Hu. Identifying authoritative and reliable contents in community question answering with domain knowledge. In *Trends and Applications in Knowledge Discovery and Data Mining*, pages 133–142. Springer, 2013.

[43] M. Gupta, R. Li, Z. Yin, and J. Han. Survey on social tagging techniques. *ACM SIGKDD Explorations Newsletter*, 12(1):58–72, 2010.

[44] Z. Gyongyi, G. Koutrika, J. Pedersen, and H. Garcia-Molina. Questioning Yahoo! Answers. Technical Report 2007-35, Stanford InfoLab, 2007.

[45] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[46] F. M. Harper, D. Moy, and J. A. Konstan. Facts or friends?: distinguishing informational and conversational questions in social Q&A sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 759–768, 2009.

[47] F. M. Harper, D. Raban, S. Rafaeli, and J. A. Konstan. Predictors of answer quality in online Q&A sites. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 865–874, 2008.

[48] D. Hiemstra and W. Kraaij. Twenty-one at trec7: Ad-hoc and cross-language track. In *Proceedings of the Seventh Text REtrieval Conference*, pages 174–185, 1998.

[49] L. Hirschman and R. Gaizauskas. Natural language question answering: The view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

[50] T. Hofmann. Probabilistic latent semantic indexing. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 50–57, 1999.

[51] D. Horowitz and S. D. Kamvar. The anatomy of a large-scale social search engine. In *Proceedings of the 19th International Conference on World Wide Web*, pages 431–440, 2010.

[52] E. Hovy and C.-Y. Lin. Automated text summarization and the SUMMARIST system. In *Proceedings of a Workshop on Held at Baltimore, Maryland: October 13-15, 1998*, pages 197–214, 1998.

[53] E. H. Hovy, L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. Question answering in webclopedia. In *Proceedings of the Ninth Text REtrieval Conference*, pages 655–664, 2000.

[54] Y. Hu, Y. Qian, H. Li, D. Jiang, J. Pei, and Q. Zheng. Mining query subtopics from search log data. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 305–314, 2012.

[55] J.-N. Hwang, S.-R. Lay, and A. Lippman. Nonparametric multivariate density estimation: a comparative study. *IEEE Transactions on Signal Processing*, 42(10):2795–2810, 1994.

[56] J. Jeon, W. B. Croft, and J. H. Lee. Finding similar questions in large question and answer archives. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*, pages 84–90, 2005.

[57] J. Jeon, W. B. Croft, J. H. Lee, and S. Park. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 228–235, 2006.

[58] Z. Ji and B. Wang. Learning to rank for question routing in community question answering. In *Proceedings of the 22nd*

ACM International Conference on Information and Knowledge Management, pages 2363–2368, 2013.

[59] P. Jurczyk and E. Agichtein. Discovering authorities in question answer communities by using link analysis. In *Proceedings of the 16th ACM International Conference on Information and Knowledge Management*, pages 919–922, 2007.

[60] W.-C. Kao, D.-R. Liu, and S.-W. Wang. Expert finding in question-answering websites: a novel hybrid approach. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 867–871, 2010.

[61] M. P. Kato, T. Sakai, and K. Tanaka. Structured query suggestion for specialization and parallel movement: Effect on search behaviors. In *Proceedings of the 21st International Conference on World Wide Web*, pages 389–398, 2012.

[62] S. Kim, J. S. Oh, and S. Oh. Best-answer selection criteria in a social Q&A site from the user-oriented relevance perspective. In *Proceeding of the 70th Annual Meeting of the American Society for Information Science and Technology*, page 4, 2007.

[63] V. Kitzie, E. Choi, and C. Shah. From bad to good: An investigation of question quality and transformation. In *Proceedings of the American Society for Information Science and Technology*, 2013.

[64] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. In *Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 668–677, 1998.

[65] O. Kucuktunc, B. B. Cambazoglu, I. Weber, and H. Ferhatosmanoglu. A large-scale sentiment analysis for Yahoo! Answers. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining*, pages 633–642, 2012.

[66] X. N. Lam, T. Vu, T. D. Le, and A. D. Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the Second International Conference on Ubiquitous Information Management and Communication*, pages 208–211, 2008.

[67] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2001.

[68] B. Li, T. Jin, M. R. Lyu, I. King, and B. Mak. Analyzing and predicting question quality in community question answering services. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 775–782, 2012.

[69] B. Li and I. King. Routing questions to appropriate answerers in community question answering services. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1585–1588, 2010.

[70] B. Li, I. King, and M. R. Lyu. Question routing in community question answering: putting category in its place. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, pages 2041–2044, 2011.

[71] B. Li, J. Liu, C.-Y. Lin, I. King, and M. R. Lyu. A hierarchical entity-based approach to structuralize user generated content in social media: A case of Yahoo! Answers. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1521–1532, 2013.

[72] B. Li, Y. Liu, and E. Agichtein. CoCQA: co-training over questions and answers with an application to predicting question subjectivity orientation. In *Proceedings of the Confer-*

ence on Empirical Methods in Natural Language Processing, pages 937–946, 2008.

[73] B. Li, Y. Liu, A. Ram, E. V. Garcia, and E. Agichtein. Exploring question subjectivity prediction in community QA. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 735–736, 2008.

[74] B. Li, M. R. Lyu, and I. King. Communities of Yahoo! Answers and Baidu Zhidao: Complementing or competing? In *Proceedings of the 2012 International Joint Conference on Neural Networks*, pages 1–8, 2012.

[75] Z. Li, H. Shen, and J. E. Grant. Collective intelligence in the online social network of Yahoo! Answers and its implications. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 455–464, 2012.

[76] C.-Y. Lin and E. Hovy. From single to multi-document summarization: a prototype system and its evaluation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 457–464, 2002.

[77] X. Ling and D. S. Weld. Fine-grained entity recognition. In *Proceedings of the 26th Conference on Artificial Intelligence*, pages 94–100, 2012.

[78] J. Liu, Y.-I. Song, and C.-Y. Lin. Competition-based user expertise score estimation. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 425–434, 2011.

[79] J. Liu, Q. Wang, C.-Y. Lin, and H.-W. Hon. Question difficulty estimation in community question answering services.

In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 85–90, 2013.

[80] M. Liu, Y. Liu, and Q. Yang. Predicting best answerers for new questions in community question answering. In *Web-Age Information Management*, volume 6184 of *Lecture Notes in Computer Science*, pages 127–138. Springer Berlin Heidelberg, 2010.

[81] Q. Liu, E. Agichtein, G. Dror, E. Gabrilovich, Y. Maarek, D. Pelleg, and I. Szpektor. Predicting web searcher satisfaction with existing community-based answers. In *Proceedings of the 34th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 415–424, 2011.

[82] T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

[83] X. Liu and W. B. Croft. Cluster-based retrieval using language models. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193, 2004.

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

[85] X. Liu, Z. Li, X. Zhao, and Z. Zhou. Using concept-level random walk model and global inference algorithm for answer summarization. In *Proceedings of the 7th Asia Conference on Information Retrieval Technology*, pages 434–445, 2011.

[86] Y. Liu and E. Agichtein. On the evolution of the Yahoo! Answers QA community. In *Proceedings of the 31st Annual In-*

ternational ACM SIGIR Conference on Research and Development in Information Retrieval, pages 737–738, 2008.

[87] Y. Liu and E. Agichtein. You've got answers: towards personalized models for predicting success in community question answering. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies*, pages 97–100, 2008.

[88] Y. Liu, J. Bian, and E. Agichtein. Predicting information seeker satisfaction in community question answering. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 483–490, 2008.

[89] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 497–504, 2008.

[90] Y. Liu, N. Narasimhan, V. Vasudevan, and E. Agichtein. Is this urgent?: exploring time-sensitive information needs in collaborative question answering. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 712–713, 2009.

[91] J. Lou, K. Lim, Y. Fang, and Z. Peng. Drivers of knowledge contribution quality and quantity in online question and answering communities. In *Proceedings of PACIS*, 2011.

[92] Y. Lu, X. Quan, X. Ni, W. Liu, and Y. Xu. Latent link analysis for expert finding in user-interactive question answering services. In *Proceedings of the Fifth International Conference on Semantics, Knowledge and Grid*, pages 54–59, 2009.

[93] H. Ma, H. Yang, M. R. Lyu, and I. King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, pages 931–940, 2008.

[94] H. Ma, D. Zhou, C. Liu, M. R. Lyu, and I. King. Recommender systems with social regularization. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, pages 287–296, 2011.

[95] R. Malouf. A comparison of algorithms for maximum entropy parameter estimation. In *Proceedings of the 6th Conference on Natural language learning*, pages 1–7, 2002.

[96] C. D. Manning, P. Raghavan, and H. Schtze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[97] P. Massa and P. Avesani. Trust-aware collaborative filtering for recommender systems. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, volume 3290 of *Lecture Notes in Computer Science*, pages 492–508. Springer Berlin Heidelberg, 2004.

[98] Q. Mei, X. Shen, and C. Zhai. Automatic labeling of multinomial topic models. In *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 490–499, 2007.

[99] E. Mendes Rodrigues and N. Milic-Frayling. Socializing or knowledge sharing?: characterizing social intent in community question answering. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, pages 1127–1136, 2009.

[100] Y. Miao, C. Li, J. Tang, and L. Zhao. Identifying new categories in community question answering archives: A topic

modeling approach. In *Proceedings of the 19th ACM Conference on Information and Knowledge Management*, pages 1673–1676, 2010.

[101] R. Mihalcea, C. Corley, and C. Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 775–780, 2006.

[102] D. R. H. Miller, T. Leek, and R. M. Schwartz. A hidden markov model information retrieval system. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 214–221, 1999.

[103] D. Moldovan, M. Paşca, S. Harabagiu, and M. Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154, 2003.

[104] D. Mollá and J. L. Vicedo. Question answering in restricted domains: An overview. *Computational Linguistics*, 33(1):41–61, 2007.

[105] D. Movshovitz-Attias, Y. Movshovitz-Attias, P. Steenkiste, and C. Faloutsos. Analysis of the reputation system and user contributions on a question answering website: StackOverflow. In *Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 886–893, 2013.

[106] K. K. Nam, M. S. Ackerman, and L. A. Adamic. Questions in, knowledge in?: a study of naver's question answering community. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 779–788, 2009.

[107] A. Y. Ng, M. I. Jordan, Y. Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in Neural Information Processing Systems*, 2:849–856, 2002.

[108] T. T. Nguyen, K. Chang, and S. C. Hui. A math-aware search engine for math question answering system. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, pages 724–733, 2012.

[109] L. Nie, B. D. Davison, and B. Wu. From whence does your authority come?: utilizing community relevance in ranking. In *Proceedings of the 22nd National Conference on Artificial intelligence*, pages 1421–1426, 2007.

[110] H. Ohshima, S. Oyama, and K. Tanaka. Searching coordinate terms with their context from the web. In *Proceedings of the 7th International Conference on Web Information Systems*, pages 40–47, 2006.

[111] A. Pal and J. A. Konstan. Expert identification in community question answering: exploring question selection bias. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, pages 1505–1508, 2010.

[112] A. Pal, F. Wang, M. X. Zhou, J. Nichols, and B. A. Smith. Question routing to user communities. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 2357–2362, 2013.

[113] V. Pande, T. Mukherjee, and V. Varma. Summarizing answers for community question answer services. In *Language Processing and Knowledge in the Web*, volume 8105 of *Lecture Notes in Computer Science*, pages 151–161. Springer Berlin Heidelberg, 2013.

[114] M. Pasca, D. Lin, J. Bigham, A. Lifchits, and A. Jain. Organizing and searching the world wide web of facts - step one: the one-million fact extraction challenge. In *Proceedings of the 21st National Conference on Artificial Intelligence*, pages 1400–1405, 2006.

[115] S. A. Paul, L. Hong, and E. H. Chi. Who is authoritative? understanding reputation mechanisms in Quora. In *Proceedings of Collective Intelligence*, 2012.

[116] M. Pennacchiotti and P. Pantel. Entity extraction via ensemble semantics. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 238–247, 2009.

[117] J. Platt et al. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, 1998.

[118] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 275–281, 1998.

[119] J. M. Prager. Open-domain question-answering. *Foundations and Trends in Information Retrieval*, 1(2):91–231, 2006.

[120] F. Provost. Machine learning from imbalanced data sets 101. In *Proceedings of the AAAI Workshop on Imbalanced Data Sets*, pages 1–3, 2000.

[121] X. Qiu, L. Tian, and X. Huang. Latent semantic tensor indexing for community-based question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, pages 434–439, 2013.

[122] M. Qu, G. Qiu, X. He, C. Zhang, H. Wu, J. Bu, and C. Chen. Probabilistic question recommendation for question answering communities. In *Proceedings of the 18th International Conference on World Wide Web*, pages 1229–1230, 2009.

[123] S. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

[124] E. M. Rodrigues, N. Milic-Frayling, and B. Fortuna. Social tagging behaviour in community-driven question answering. In *Proceedings of IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 1, pages 112–119, 2008.

[125] G. Salton and M. J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc., 1986.

[126] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613–620, 1975.

[127] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock. Methods and metrics for cold-start recommendations. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 253–260, 2002.

[128] C. Shah, S. Oh, and J. S. Oh. Research agenda for social Q&A. *Library & Information Science Research*, 31(4):205–209, 2009.

[129] C. Shah and J. Pomerantz. Evaluating and predicting answer quality in community QA. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 411–418, 2010.

[130] D. Shahaf, J. Yang, C. Suen, J. Jacobs, H. Wang, and J. Leskovec. Information cartography: creating zoomable, large-scale maps of information. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1097–1105, 2013.

[131] B. Shapira. *Recommender systems handbook*. Springer, 2011.

[132] A. Shepitsen, J. Gemmell, B. Mobasher, and R. Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM Conference on Recommender Systems*, pages 259–266, 2008.

[133] S. Shi, H. Zhang, X. Yuan, and J.-R. Wen. Corpus-based semantic class mining: distributional vs. pattern-based approaches. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 993–1001, 2010.

[134] X. Si, E. Y. Chang, Z. Gyöngyi, and M. Sun. Confucius and its intelligent disciples: Integrating social with search. *Proceedings of the VLDB Endowment*, 3(1-2):1505–1516, 2010.

[135] A. Singh. Entity based QA retrieval. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1266–1277, 2012.

[136] A. Singh. Entity based translation language model. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 599–600, 2012.

[137] Y.-I. Song, C.-Y. Lin, Y. Cao, and H.-C. Rim. Question utility: A novel static ranking of question search. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, pages 1231–1236, 2008.

[138] M. A. Suryanto, E. P. Lim, A. Sun, and R. H. L. Chiang. Quality-aware collaborative question answering: methods and evaluation. In *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, pages 142–151, 2009.

[139] I. Szpektor, Y. Maarek, and D. Pelleg. When relevance is not enough: Promoting diversity and freshness in personalized question recommendation. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1249–1260, 2013.

[140] H. Toba, M. Zhao-Yan, M. Adrianz, and C. T. Seng. Discovering high quality answers in community question answering archives using a hierarchy of classifiers. *Information Sciences*, 2013.

[141] M. Tomasoni and M. Huang. Metadata-aware measures for answer summarization in community question answering. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 760–769, 2010.

[142] G. Wang, K. Gill, M. Mohanlal, H. Zheng, and B. Y. Zhao. Wisdom in the social crowd: An analysis of Quora. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1341–1352, 2013.

[143] J. Wang, A. P. de Vries, and M. J. T. Reinders. Unifying user-based and item-based collaborative filtering approaches by similarity fusion. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 501–508, 2006.

[144] K. Wang, Z. Ming, and T.-S. Chua. A syntactic tree matching approach to finding similar questions in community-based QA services. In *Proceedings of the 32nd Annual International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pages 187–194, 2009.

[145] K. Wang, Z.-Y. Ming, X. Hu, and T.-S. Chua. Segmentation of multi-sentence questions: towards effective question retrieval in cQA services. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 387–394, 2010.

[146] X.-J. Wang, X. Tu, D. Feng, and L. Zhang. Ranking community answers by modeling question-answer relationships via analogical reasoning. In *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 179–186, 2009.

[147] B. Webber and N. Webb. *Question Answering*, pages 630–654. Wiley-Blackwell, 2010.

[148] H. Wenwen, Q. Xirong, S. Siqi, T. Ye, and W. Wendong. Ranking potential reply-providers in community question answering system. *China Communications*, 10(10):125–136, 2013.

[149] H. Xuan, Y. Yang, and C. Peng. An expert finding model based on topic clustering and link analysis in CQA website. *Journal of Network & Information Security*, 4(2):165–176, 2013.

[150] X. Xue, J. Jeon, and W. B. Croft. Retrieval models for question and answer archives. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–482, 2008.

[151] J. Yamron. Topic detection and tracking segmentation task. In *Proceedings of the Topic Detection and Tracking Workshop*, 1997.

[152] L. Yang, S. Bao, Q. Lin, X. Wu, D. Han, Z. Su, and Y. Yu. Analyzing and predicting not-answered questions in community-based question answering services. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*, pages 1273–1278, 2011.

[153] S. Yang, E. K. Lua, and Y. Wang. Towards human-centric personalized expertise ranking in community-based question answering. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking*, pages 41–46, 2013.

[154] S. R. Yerva, Z. Miklos, F. Grosan, A. Tandrau, and K. Aberer. Tweetspector: Entity-based retrieval of tweets. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1016–1016, 2012.

[155] Y. Yokoyama, T. Hochin, and H. Nomiya. Estimation of objective scores of answer statements posted at Q&A sites. In *The 14th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing*, pages 426–431, 2013.

[156] Q. Yuan, G. Cong, A. Sun, C.-Y. Lin, and N. M. Thalmann. Category hierarchy maintenance: A data-driven approach. In *Proceedings of the 35th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 791–800, 2012.

[157] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342, 2001.

[158] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22:179–214, 2004.

[159] H. Zhang, M. Zhu, S. Shi, and J.-R. Wen. Employing topic models for pattern-based semantic class discovery. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 459–467, 2009.

[160] J. Zhang, M. S. Ackerman, and L. Adamic. Expertise networks in online communities: structure and algorithms. In *Proceedings of the 16th International Conference on World Wide Web*, pages 221–230, 2007.

[161] J. Zhang, M. S. Ackerman, L. Adamic, and K. K. Nam. QuME: A mechanism to support expertise finding in online help-seeking communities. In *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, pages 111–114, 2007.

[162] Z.-K. Zhang, C. Liu, Y.-C. Zhang, and T. Zhou. Solving the cold-start problem in recommender systems with social tags. *Europhysics Letters*, 92(2):28002:1–28002:6, 2010.

[163] G. Zhou, Y. Chen, D. Zeng, and J. Zhao. Towards faster and better retrieval models for question search. In *Proceedings of the 22nd ACM International Conference on Information and Knowledge Management*, pages 2139–2148, 2013.

[164] L. Zhou, C. Y. Lin, and E. Hovy. Summarizing answers for complicated questions. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation*, pages 737–740, 2006.

[165] T. C. Zhou, M. R. Lyu, and I. King. A classification-based approach to question routing in community question answering. In *Proceedings of the 21st International Conference Companion on World Wide Web*, pages 783–790, 2012.

[166] T. C. Zhou, X. Si, E. Y. Chang, I. King, and M. R. Lyu. A data-driven approach to question subjectivity identification in community question answering. In *Proceedings of AAAI Conference on Artificial Intelligence*, pages 164–170, 2012.

[167] Y. Zhou, G. Cong, B. Cui, C. S. Jensen, and J. Yao. Routing questions to the right users in online communities. In *Proceedings of the 2009 IEEE International Conference on Data Engineering*, pages 700–711, 2009.

[168] X. Zhu. *Semi-Supervised Learning with Graphs*. PhD thesis, Carnegie Mellon University, 2005.

[169] X. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of International Conference on Machine Learning*, pages 912–919, 2003.

[170] X. Zhu, Z.-Y. Ming, X. Zhu, and T.-S. Chua. Topic hierarchy construction for the organization of multi-source user generated contents. In *Proceedings of the 36th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 233–242, 2013.

[171] Z. Zhu, D. Bernhard, and I. Gurevych. *A Multi-Dimensional model for assessing the quality of answers in social Q&A sites*. PhD thesis, Technische Universität Darmstadt, 2009.

[172] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Computing Surveys*, 38(2), 2006.