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：李晸
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+和百度知道等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。為了方便回答者找到合適的問題，幫助提問者更有效的獲取資訊，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，利用問隙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。為了解算回答者的專業知識，我們提出了三個模型。第一個模型來自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去對所有回答者進行估計，效率不高又費時。為了解決這個問題，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構建社區問題服務中的問題。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。為了解決這個問題，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中知識學習的表現比基於列表的顯著提高。在系統層面上，實體樹通過再排序顯著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為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

Abstract

Acknowledgement

1 Introduction

1.1 Overview ........................................ 1
1.2 Thesis Contribution ............................ 7
1.3 Thesis Organization ......................... 9

2 Background Review

2.1 Yahoo! Answers ................................. 12
2.2 Studies in CQA Services ..................... 13
2.3 Question Retrieval ............................. 16
  2.3.1 Basic Models ............................... 16
  2.3.2 Category Information .................... 19
  2.3.3 Syntactic Tree Matching ................. 20
  2.3.4 Segmentation-aided Retrieval ............ 21
  2.3.5 Answer Quality ............................ 22
  2.3.6 Latent Semantic Tensor Indexing ......... 22
2.4 Question Classification ...................... 23
2.5 Question Routing .............................. 25
  2.5.1 Topic Model-based QR .................... 26
  2.5.2 Language Model-based QR ............... 27
  2.5.3 Classification-based QR .................. 29
  2.5.4 Diversity and Freshness (DF model) .... 31
| 2.5.5 | Model Comparison | 31 |
| 2.6 | Answer Quality Evaluation | 33 |
| 2.6.1 | Features | 34 |
| 2.6.2 | Models | 35 |
| 2.7 | Answer Summarization | 37 |
| 2.7.1 | Constraint-based Summarization | 38 |
| 2.7.2 | Question Type-based Summarization | 40 |
| 2.7.3 | Graph-based Summarization | 41 |
| 2.7.4 | Summary | 42 |

3 Question Popularity Analysis and Prediction | 44 |
| 3.1 | Motivation and Problem | 44 |
| 3.2 | Data Description | 46 |
| 3.2.1 | Data Set | 46 |
| 3.2.2 | Ground Truth Setting | 47 |
| 3.3 | Factors Affecting QP | 48 |
| 3.4 | Prediction of QP | 50 |
| 3.4.1 | Algorithm | 52 |
| 3.4.2 | Experimental Setup | 56 |
| 3.4.3 | Experimental Results | 59 |
| 3.5 | Summary | 62 |

4 Question Routing | 64 |
| 4.1 | Problem and Motivation | 64 |
| 4.2 | Question Routing Framework | 65 |
| 4.2.1 | Answerer Profiling | 66 |
| 4.2.2 | Expertise Estimation | 67 |
| 4.2.3 | Availability Estimation | 72 |
| 4.2.4 | Answerer Ranking | 73 |
| 4.3 | Experimental Setup | 74 |
| 4.3.1 | Data Collection | 75 |
| 4.3.2 | Methods for Comparison | 77 |
| 4.3.3 | Evaluation Metrics | 79 |
# Conclusion

## Summary

## Future Work

| Bibliography | 138 | 134 | 136 |
# List of Figures

1.1 The computational framework for question processing in CQA services. 3

2.1 A question’s life in Yahoo! Answers. 13

2.2 Research topics of CQA services 15

3.1 Construct of question popularity in CQA. 45

3.2 Distributions of QP in two topics. Left: *Music*; Right: *Movies*. 49

3.3 A toy example. Left: askers; Right: questions in three topics. 52

3.4 Sensitivity versus training rate across various methods in *Music*. 61

3.5 Specificity versus training rate across various methods in *Music*. 62

4.1 The framework of *Question Routing*. 66

4.2 The MRR value of Basic Q and Smoothed Q versus various $\alpha$. 82

4.3 The MRR value of Smoothed Q versus various $\beta$. 83

4.4 MRR versus $\gamma$ across different methods. 84

5.1 An example of question category in CQA services (captured from Yahoo! Answers on January 20, 2011). 87

5.2 Question category for expertise estimation in QR. 88

5.3 An example of Matrix $E$. 92
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). ................................................. 92
5.5 MRR for TCS-LM using answerer-based and content-based approaches to estimate transferring probability under GT-A. ................................................................. 107
5.6 Prec@K of LM with different answerer priors. ...... 108
5.7 Prec@K of BCS-LM with different answerer priors. 109
6.1 An CET constructed from questions about Edinburgh. 113
6.2 The interface of CET-based program. ................. 126
6.3 Re-ranking results of Computer & Internet. ....... 131
6.4 Re-ranking results of Travel ............................ 132
## List of Tables

1.1 Comparison between CQA and QA. .......................... 2

2.1 Basic models for question retrieval. .......................... 16

2.2 Notations used in Section 2.3. .......................... 17

2.3 Comparison among different approaches for question routing. .......................... 30

2.4 Comparison among different answer summarization approaches. .......................... 43

3.1 Summary of questions and askers in *Entertainment & Music* category and its subcategories. .......................... 46

3.2 Rule base for ground truth setting. .......................... 48

3.3 Summary of questions in four levels. .......................... 48

3.4 Summary of QP for different askers. .......................... 50

3.5 Summary of data in Study 2. .......................... 56

3.6 Summary of features extracted from questions and askers. .......................... 57

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*: $\alpha = 0.8$, $\beta = 0.1$). .......................... 60

4.1 Statuses of tracked questions two days after being posted. .......................... 65
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 \( q_r \). ................. 70

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 \( q_r \). .... 71

4.4 Description of our data set. ................. 76

4.5 Number of successful QR and Prec@\( K \) versus \( K \) for QLLM. ................. 81

4.6 Different methods’ MRR in QR. ................. 81

5.1 Description of our data set (after stop words removing and stemming). ................. 93

5.2 Transferred probabilities between partial leaf categories (answerer-based method). ................. 99

5.3 Transferred probabilities between partial leaf categories (content-based method). ................. 100

5.4 Effects of using category-answerer indexes on answerer filtering. ................. 101

5.5 Different methods’ Prec@\( K \) in QR versus various \( K \)'s using GT-A (best results are shown in bold). .... 102

5.6 Different methods’ Prec@\( K \) in QR versus various \( K \)'s using GT-BA (best results are shown in bold). .... 103

5.7 Different methods’ MQRT in QR (in seconds). .... 104

5.8 MRR and MAP of various models under GT-A (best results are shown in bold). ................. 105

5.9 MRR and MAP of various models under GT-BA (best results are shown in bold). ................. 106

6.1 Sample questions about Edinburgh. ................. 112
6.2 Category mapping between Yahoo! Answers and FreeBase.

6.3 Number of questions and entities in Set EC.

6.4 Entity extraction for various methods.

6.5 Clustering results of Cars & Transportation and Computer & Internet using AC-MAX ($\theta_{max}=0.1$).

6.6 Clustering results of Sports and Travel using AC-MAX ($\theta_{max}=0.1$).

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

6.8 User study results.

6.9 Questionnaire results.

6.10 Re-ranking results for VSM and QLLM (* means that $p < 0.05$ in students’ t-test).
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, launched 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/
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 Google⁶ and Bing⁷ 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.

---

⁶http://www.google.com/
⁷http://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\textsuperscript{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-\footnote{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 1. INTRODUCTION

• **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
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
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.

<table>
<thead>
<tr>
<th>Model</th>
<th>Paper</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vector space model</td>
<td>[32, 19]</td>
</tr>
<tr>
<td>Okapi BM25 model</td>
<td>[56, 19]</td>
</tr>
<tr>
<td>Query likelihood language model</td>
<td>[56, 32, 21, 19]</td>
</tr>
<tr>
<td>Translation model</td>
<td>[32, 150, 19]</td>
</tr>
<tr>
<td>Translation-based language model</td>
<td>[150, 19]</td>
</tr>
</tbody>
</table>

### 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:

$$Sim_{VSM}(q, d) = \frac{\sum_{t \in q \cap d} w_{q,t} w_{d,t}}{W_q W_d}.$$  \hspace{1cm} (2.1)
Table 2.2: Notations used in Section 2.3.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q$</td>
<td>a query</td>
</tr>
<tr>
<td>$d$</td>
<td>a question</td>
</tr>
<tr>
<td>$a$</td>
<td>an answer</td>
</tr>
<tr>
<td>$t$</td>
<td>a term</td>
</tr>
<tr>
<td>Coll</td>
<td>the whole collection of questions</td>
</tr>
<tr>
<td>$N$</td>
<td>the number of questions in the whole collection</td>
</tr>
<tr>
<td>$df_t$</td>
<td>the number of questions containing the term $t$</td>
</tr>
<tr>
<td>$tf_{q,t}$</td>
<td>the frequency of term $t$ in $q$</td>
</tr>
<tr>
<td>$tf_{t,d}$</td>
<td>the frequency of term $t$ in $d$</td>
</tr>
<tr>
<td>$tf_{t,Coll}$</td>
<td>the frequency of term $t$ in the whole collection</td>
</tr>
<tr>
<td>$Sim(q, d)$</td>
<td>similarity score between a query $q$ and a question $d$</td>
</tr>
</tbody>
</table>

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].

\[ Sim_{BM25}(q, d) = \sum_{t \in q \cap d} w_{q,t} w_{d,t}, \]  

(2.6)

where

\[ w_{q,t} = \ln\left(\frac{N - df_t + 0.5}{df_t + 0.5}\right) \frac{(k_3 + 1) tf_t,q}{k_3 + tf_t,q}, \]  

(2.7)

\[ w_{d,t} = \frac{(k_1 + 1) tf_t,d}{K_d + tf_t,d}, \]  

(2.8)

\[ K_d = k_1(1 - b + b \frac{W_d}{W_A}). \]  

(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:

\[ Sim_{QLLM}(q, d) = \prod_{t \in q} (1 - \lambda) P_{ml}(t|d) + \lambda P_{ml}(t|Coll), \]  

(2.10)

where

\[ P_{ml}(t|d) = \frac{tf_t,d}{\sum_{t' \in d} tf_{t',d}}, \]  

(2.11)

\[ P_{ml}(t|Coll) = \frac{tf_t,Coll}{\sum_{t' \in Coll} tf_{t',Coll}}. \]  

(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 suffer 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],

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

(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,

\[
Sim_{TBLM}(q,d) = \prod_{t \in q} (1 - \lambda)(\beta \sum_{w \in d} T(t|w)P_{ml}(w|d)) + (1 - \beta)P_{ml}(t|d) + \lambda P_{ml}(t|Coll),
\]

(2.14)

where \( \beta \) 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

\[
Sim_{c_1}(q, d) = (1 - \beta)N(S_{q,d}) + \beta N(S_{q,cat(d)}),
\]

(2.15)

where \( \beta \) 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:

\[
Sim_{STM}(q, d) = \frac{S(T_1, T_2)}{\sqrt{S(T_1, T_1)S(T_2, T_2)}},
\]

(2.16)

\[
S(T_1, T_2) = \sum_{r_1 \in T_1} \sum_{r_2 \in T_2} M(r_1, r_2),
\]

(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:

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

(2.18)
In the above equation, $\delta_{r_1}$ 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. $\lambda$ and $\mu$ are two parameters denoting the preference between size and depth, $\eta$ 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,

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

(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:

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

where \( \tau \) 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 \((\text{question title, question content, answer content})\), 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 \(\langle qt_i, qc_i, a_i \rangle (i = 1, \ldots, N)\), a tensor \(D \in \mathbb{R}^{N \times 3 \times T}\) is utilized to represent the collection, where \(T\) is the number of terms. In \(D\), the three dimensions correspond to entries, parts, and terms respectively.

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

\[
Sim_{LSTI}(q, d) = \frac{\sum_{i,j} \hat{D}_{q_{i,j}} \hat{D}_{d_{i,j}}}{\sum_{i,j} \hat{D}_{q_{i,j}}^2 \times \sum_{i,j} \hat{D}_{d_{i,j}}^2}.
\] (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. Empirical results show that the two classifiers performed similarly. In addition, question text
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 $w$:

$$P(u, w) = \sum_z P(u|z)P(w|z)P(z), \quad (2.22)$$

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:

$$P(u, q) = \prod_i P(u, w_i), \quad (2.23)$$

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.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Topic model</td>
<td></td>
<td>UQA</td>
<td>UQA</td>
<td>term-level similarity and topic-level similarity</td>
<td>all users and questions</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>[41]</td>
<td>UQA</td>
<td>UQA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>[122]</td>
<td>PLSA</td>
<td>PLSA</td>
<td>joint user and question probability over topics</td>
<td>all users and questions</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Language model</td>
<td></td>
<td>word generation probability estimation</td>
<td>word generation probability estimation</td>
<td>language models</td>
<td>single user and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>[167]</td>
<td>word generation probability estimation and LDA</td>
<td>word generation probability estimation and LDA</td>
<td>language models (with user authority and user availability)</td>
<td>all users and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>word generation probability estimation and LDA</td>
<td>word generation probability estimation and LDA</td>
<td>language models (with user authority and user availability)</td>
<td>all users and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>[165]</td>
<td>feature extraction</td>
<td>feature extraction</td>
<td>SVM</td>
<td>single user and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td>[29]</td>
<td>feature extraction</td>
<td>feature extraction</td>
<td>GBDT</td>
<td>single user and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Classification-based model</td>
<td></td>
<td>feature extraction</td>
<td>feature extraction</td>
<td>SVM</td>
<td>single user and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>feature extraction</td>
<td>feature extraction</td>
<td>GBDT</td>
<td>single user and single question</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>DF model</td>
<td></td>
<td>three-level probability trees</td>
<td>LDA, lexical, and category features</td>
<td>vector similarity and proactive diversification</td>
<td>single user and single question</td>
<td>yes</td>
<td>no</td>
</tr>
<tr>
<td></td>
<td>[139]</td>
<td>three-level probability trees</td>
<td>LDA, lexical, and category features</td>
<td>vector similarity and proactive diversification</td>
<td>single user and single question</td>
<td>yes</td>
<td>no</td>
</tr>
</tbody>
</table>
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 nonsence 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 were first labeled by Mechanical Turk 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.


3https://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:

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

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. \(\lambda\) are parameters of the maximum entropy model, which can be estimated from [95]. In addition,

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

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
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:

$$
\text{score}(a) = r\text{score}(q, a) \cdot q\text{socre}_{\langle\text{model}\rangle}(q, a),
$$

(2.26)

where $r\text{score}(q, a)$ measures the relevance between $q$ and $a$, which can be obtained through the query likelihood retrieval model. In addition, $q\text{socre}(q, a)$ represents the quality of $a$, which is derived from the asking expertise ($e_{\text{ask}}$) and answering expertise ($e_{\text{ans}}$) of its answerer $u_i$:

$$
q\text{socre}_{\langle\text{model}\rangle}(q, a) = \sigma \cdot e_{\text{ask}}(u_i, q) + (1 - \sigma) \cdot e_{\text{ans}}(u_i, q),
$$

(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.
CHAPTER 2. BACKGROUND REVIEW

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, \( \text{length}(j) \) 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:

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

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,

\[
S(c_i) = (1 - d) \cdot \sum_{c_k \in C} \frac{\text{sim}(c_i, c_k)}{\sum_{c_j \in C} \text{sim}(c_i, c_j)} \cdot p(c_k|q) \\
+ d \cdot \frac{\text{rel}(c_i|q)}{\sum_{c_j \in C} \text{rel}(c_j|q)}
\]  

(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 \( q \), \( \text{rel}(c_i|q) \) represents the maximum similarity between \( c_i \) and all concepts in \( q \), and \( \text{sim}(c_i, c_k) \) denotes
the similarity between two concepts \( c_i \) and \( c_k \) \[101\]. In the second model, user authority is incorporated:

\[
S(c_i) = (1 - d) \cdot \sum_{c_k \in C} \frac{\text{sim}(c_i, c_k | u_i, u_k)}{\sum_{c_j \in C} \text{sim}(c_i, c_j | u_i, u_j)} \cdot p(c_k | q)
+ d \cdot \frac{\text{rel}(c_i | q)}{\sum_{c_j \in C} \text{rel}(c_j | q)},
\]

(2.29)

where

\[
\text{sim}(c_i, c_j | u_i, u_j) = \text{sim}(c_i, c_j | u_i) + \text{sim}(c_i, c_j | u_j)
= \text{sim}(c_i, c_j) \cdot \text{auth}(u_i)
+ \text{sim}(c_i, c_j) \cdot \text{auth}(u_j).
\]

(2.30)

In the above equation, \( \text{auth}(u_i) \) 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 $x$ denote the sentence sequence from all answers to a question, $y$ be the corresponding labels, if $y_i = 1$, $x_i$ will be incorporated in the summarized answer. According to CRF, the conditional probability of $y$ given $x$ is:

$$
p(y|x) = \frac{1}{Z(x)} \exp\left(\sum_{v \in V,l} \mu_l g_l(v, y|v, x) + \sum_{e \in E,k} \lambda_k f_k(e, y|e, x)\right),
$$

where $V$ and $E$ represent the node set and the edge set, respectively. In addition, $\frac{1}{Z(x)}$ 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), $\mu$ and $\lambda$ are weights, and $y|v$ ($y|e$) denotes the component of $y$ 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:

$$\text{maximize} \sum_{i=1}^{N} \log(p_{\theta}(y^i|x^i)) - C \sum_{g=1}^{G} \alpha_g$$

subject to $\alpha_g \geq \|\tilde{\theta}_g\|_2, \forall g$.

In the above optimization problem, $G$ denotes the number of feature groups, $\tilde{\theta}_g$ 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 $x$:

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

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.

<table>
<thead>
<tr>
<th>Approach</th>
<th>Completeness</th>
<th>Concision</th>
<th>Summation</th>
<th>Readability</th>
<th>Quality</th>
</tr>
</thead>
<tbody>
<tr>
<td>Constraint-based</td>
<td>Good</td>
<td>Constrain dependent</td>
<td>Sentence level</td>
<td>Not considered</td>
<td>Good</td>
</tr>
<tr>
<td>Question type-based</td>
<td>Good</td>
<td>Type dependent</td>
<td>Answer and sentence level</td>
<td>Good</td>
<td>Not considered</td>
</tr>
<tr>
<td>Graph-based</td>
<td>Good</td>
<td>Not considered</td>
<td>Sentence level</td>
<td>Not considered</td>
<td>Good</td>
</tr>
</tbody>
</table>

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

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

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.

<table>
<thead>
<tr>
<th>RM</th>
<th>NTA</th>
<th>4</th>
<th>3</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td></td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td></td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td></td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3.3: Summary of questions in four levels.

<table>
<thead>
<tr>
<th>Level</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>53,806</td>
<td>62,192</td>
<td>69,836</td>
<td>52,715</td>
</tr>
</tbody>
</table>

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.
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 important 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.

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

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 ($q_1, \ldots, q_5$) in three topics ($t_1, t_2, \text{ and } t_3$). 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, 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 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.
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 $E'$:

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

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 $x_i$ and $x_j$:

$$w(q_i, q_j) = \exp\left(-\frac{||x_i - x_j||^2}{\lambda_q^2}\right), \quad (3.2)$$

where $\lambda_q$ 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 $w(q_i, q_i) = 0$. 

Figure 3.3: A toy example. Left: askers; Right: questions in three topics.
Algorithm 1 MRLP

Input: asker expertise vector $U_0^k$, QP vector $Q_0^k$, $E$, transition matrixes $M$ and $N$, weighting coefficients $\alpha$ and $\beta$, 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 $N$:

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

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 $\lambda_a$ 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., $Q^k = [\hat{q}_1^k, \ldots, \hat{q}_l^k, \hat{q}_{(l+1)}^k, \ldots, \hat{q}_{(l+u)}^k]^T$ and $U^k = [u_1^k, \ldots, u_x^k, u_{(x+1)}^k, \ldots, u_{(x+y)}^k]^T$. Thus, We can split $E'$, $E^T$, $M$ and $N$
into four parts:

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

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

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

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

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:

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

Let

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

we get

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

\[
\sum_j A_{ij}^n = \sum_j \sum_k A_{ik}^{n-1} A_{kj},
\]

\[
= \sum_k A_{ik}^{n-1} \sum_j A_{kj},
\]

\[
\leq \sum_k A_{ik}^{n-1} \gamma,
\]

\[
\leq \gamma^n,
\]

which means the sum of each row of \( A \) converges to zero. Therefore,

\[
A^n \begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix}_0 \rightarrow 0.
\]

Finally, we get

\[
\begin{bmatrix} U^k_y \\ Q^k_u \end{bmatrix} = (I - A)^{-1} b,
\]

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.

<table>
<thead>
<tr>
<th>Item</th>
<th>Music</th>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of questions</td>
<td>7,373</td>
<td>1,076</td>
</tr>
<tr>
<td>Number of high-popularity questions</td>
<td>3,670</td>
<td>331</td>
</tr>
<tr>
<td>Number of low-popularity questions</td>
<td>3,703</td>
<td>745</td>
</tr>
<tr>
<td>Number of askers</td>
<td>314</td>
<td>56</td>
</tr>
</tbody>
</table>

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.

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

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:

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

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:

$$f_u = (D_{uu} - W_{uu})^{-1}W_{ul}f_l = (I - P_{uu})^{-1}P_{ul}f_l. \quad (3.5)$$
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: $\alpha = 0.8$, $\beta = 0.1$).

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy under training rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Music</td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td>LR_Q</td>
<td>0.550</td>
</tr>
<tr>
<td>LR_QA</td>
<td>0.567</td>
</tr>
<tr>
<td>HF_Q</td>
<td>0.535</td>
</tr>
<tr>
<td>HF_QA</td>
<td>0.547</td>
</tr>
<tr>
<td>SGBT_Q</td>
<td>0.560</td>
</tr>
<tr>
<td>SGBT_QA</td>
<td>0.576</td>
</tr>
<tr>
<td>MRLP</td>
<td>0.608</td>
</tr>
</tbody>
</table>

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.

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.

In particular, 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: Statues of tracked questions two days after being posted.

<table>
<thead>
<tr>
<th>CQA portal</th>
<th>Number of resolved questions</th>
<th>Number of unresolved but unanswered questions</th>
<th>Number of unanswered questions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yahoo! Answers</td>
<td>527</td>
<td>1,820</td>
<td>442</td>
</tr>
<tr>
<td>Baidu Zhidao</td>
<td>682</td>
<td>1,325</td>
<td>993</td>
</tr>
</tbody>
</table>

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 question 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*...
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 $q_{u_i}$:

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

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

With Jelinek-Mercer smoothing [158],

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

where $C$ is the collection of all questions, and $\lambda$ 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].

\[ P_{ml}(\omega|q_{ui}) = \frac{tf(\omega, q_{ui})}{\sum_{\omega' \in q_{ui}} tf(\omega', q_{ui})}, \]  

(4.4)

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

\[ P_{ml}(\omega|C) = \frac{tf(\omega, C)}{\sum_{\omega' \in C} tf(\omega', C)}, \]  

(4.5)

where \( tf(\omega, C) \) is the term frequency of the term \( \omega \) 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,

\[ E(u_i, q_r) = \alpha \cdot P(q_r|q_{ui}) + (1 - \alpha) \cdot Q(u_i, q_r), \]  

(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:

\[
Q_{BM}(u_i, q_r) = \frac{\sum_{q_j \sim u_i} Q(u_i, q_j) \cdot \text{sim}(q_j, q_r)}{\sum_{q_j \sim u_i} \text{sim}(q_j, q_r)},
\]

(4.7)

where \(q_j \sim u_i\) denotes the questions \(u_i\) has answered, and \(\text{sim}(q_j, q_r)\) 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:

\[
w_{qt} = \text{tf}_{t,q} \times \log \frac{N}{df_t},
\]

(4.8)
in which \(\text{tf}_{t,q}\) 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:

\[
\text{sim}(a, b) = \frac{\sum_{i=1}^{T} w_{ai} \cdot w_{bi}}{\sqrt{\sum_{i=1}^{T} w_{ai}^2} \sqrt{\sum_{i=1}^{T} w_{bi}^2}}.
\]

(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,

\[
Q_{BM}(u_1, q_r) = \frac{Q(u_1, q_2) \cdot \text{sim}(q_2, q_r)}{\text{sim}(q_2, q_r)} = Q(u_1, q_2).
\]

(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 \(q_r\).

\[
\begin{array}{|c|c|c|c|c|}
\hline
\text{Answerer} & q_1 & q_2 & q_3 & q_4 & q_r \\
\hline
u_1 & 0.7 & ? & & & \\
\hline
u_2 & 0.5 & & & & \\
\hline
u_3 & 0.9 & 0.8 & & & \\
\hline
u_4 & & 0.6 & & & \\
\hline
\end{array}
\]

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 \cite{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,

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

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 $q_r$.

<table>
<thead>
<tr>
<th>Answerer</th>
<th>Question</th>
<th>$q_1$</th>
<th>$q_2$</th>
<th>$q_3$</th>
<th>$q_4$</th>
<th>$q_r$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_1$</td>
<td></td>
<td>0.7</td>
<td>?</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_2$</td>
<td></td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_3$</td>
<td></td>
<td>0.9</td>
<td>0.8</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$u_4$</td>
<td></td>
<td></td>
<td></td>
<td>0.6</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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 $a_{ij}$ 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 $P(a_{ij})$, we use logistic regression to model the
CHAPTER 4. QUESTION ROUTING

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

\[
\log\left(\frac{P(a_{ij})}{1 - P(a_{ij})}\right) = \sigma^T a_{ij},
\]

(4.13)

where \(\sigma\) 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 by 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 logs 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:

$$A(u_i, t) = \lambda_1 A(u_i, t - 1) + \ldots + \lambda_p A(u_i, t - p) + \varepsilon,$$  \hspace{1cm} (4.14)

where $p$ is a time parameter, $\varepsilon$ is the source of randomness, and $\lambda$s 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 $\{A(u_i, t), A(u_i, t - 1), \ldots, A(u_i, t - p)\}$ ($i = 1, \ldots, m$) where $m$ is the number of answerers, we can estimate the value of $\lambda_1, \lambda_2, \ldots, \lambda_p$. Then we can apply the above model to predict the value of $A(u_i, t)$ when given $A(u_i, t - 1), \ldots, A(u_i, t - p)$.

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

$$A(u_i, T) = 1 - \prod_{j=1}^{s} (1 - A(u_i, t_j)).$$  \hspace{1cm} (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:

$$QR(u_i, q_r, T) = \gamma \cdot E(u_i, q_r) + (1 - \gamma) \cdot A(u_i, T),$$  \hspace{1cm} (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 \( T \), \( K \), \( \alpha \), \( \beta \), and \( \gamma \).

**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.

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

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 Smoothe 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:

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

- **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:

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

- **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:

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

- **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:

$$QR(u_i, q_r, T) = \gamma \cdot [\alpha \cdot P(q_r|q_u_i) + (1 - \alpha) \cdot Q_{SM}(u_i, q_r)] + (1 - \gamma) \cdot A(u_i, T). \quad (4.22)$$
4.3.3 Evaluation Metrics

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

- **Precision at K** $(\text{Prec} @ K)$. For a set of new questions $Q_r$, $\text{Prec} @ K$ 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. $\text{Prec} @ K$ is calculated as:

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

(4.23)

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

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

(4.25)

where $\text{Rank}(q_r)$ 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 $K = \text{ALL}$, 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 $\alpha$ is 0.6 for those methods which consider answer quality. The value of $\beta$ 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.

<table>
<thead>
<tr>
<th>K</th>
<th>5</th>
<th>10</th>
<th>20</th>
<th>40</th>
<th>60</th>
<th>80</th>
<th>100</th>
<th>1000</th>
<th>ALL (16,298)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of Successful QR</td>
<td>80</td>
<td>121</td>
<td>175</td>
<td>253</td>
<td>312</td>
<td>350</td>
<td>386</td>
<td>876</td>
<td>1,713</td>
</tr>
<tr>
<td>Prec@K(%)</td>
<td>3.12</td>
<td>4.72</td>
<td>6.83</td>
<td>9.87</td>
<td>12.17</td>
<td>13.65</td>
<td>15.05</td>
<td>34.17</td>
<td>66.81</td>
</tr>
</tbody>
</table>

Table 4.6: Different methods’ MRR in QR.

<table>
<thead>
<tr>
<th>QLL</th>
<th>Basic Q</th>
<th>Smoothed Q</th>
<th>QLL+AE</th>
<th>Basic Q+AE</th>
<th>Smoothed Q+AE</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.0389</td>
<td>0.0494</td>
<td>0.052</td>
<td>0.0405</td>
<td>0.0511</td>
<td>0.0541</td>
</tr>
</tbody>
</table>

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 β = 0.8 and tested different settings of α. The results are reported in Figure 4.2.

First we observe that when α = 0.6, both Basic Q and Smoothed Q attain the highest MRR. Furthermore, when α > 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.
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 $\beta$ ($\alpha = 0.6$), from which we find that the value of $\beta$ 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 $\beta = 0.8$. 
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 $\gamma$ 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-
Previously 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
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-
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 \( q_{ui} \) denote all previously answered questions by answerer \( u_i \) (i.e., \( u_i \)'s profile), and \( C = \{c_1, c_2, c_3, \ldots, c_n\} \) 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:

\[
E(u_i, q_r, c_j) \equiv P_{bc}(u_i|q_r, c_j), \tag{5.1}
\]

\[
P_{bc}(u_i|q_r, c_j) \propto P_{bc}(q_r, c_j|u_i)P(u_i), \tag{5.2}
\]

\[
P_{bc}(q_r, c_j|u_i) = P_{bc}(q_r|c_j, u_i)P(c_j|u_i), \tag{5.3}
\]

\[
P_{bc}(q_r|c_j, u_i) = P_{bc}(q_r|c_j, q_{ui}) = \prod_{\omega \in q_r} P(\omega|q_{ui}), \tag{5.4}
\]

and

\[
P(\omega|q_{ui}) = (1 - \lambda)P_{ml}(\omega|q_{ui}) + \lambda P_{ml}(\omega|Coll), \tag{5.5}
\]

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 \( q_{ui} \) 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:

\[
P_{tcs}(q_r, c_j | u_i) = \beta P_{bcs}(q_r, c_j | u_i) + \sum_{c_k \in Tran(c_j)} T(c_k \rightarrow c_j) P_{bcs}(q_r, c_k | u_i)
\]

where \( \beta \) adjusts the weight between the original leaf category and other similar leaf categories. Lower \( \beta \) 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 \( T(c_k \rightarrow c_j) \) represents the probability of transfer from category \( c_k \) to \( c_j \). Furthermore, we define

\[
c_k \in Tran(c_j) \text{ if } T(c_k \rightarrow c_j) \geq \delta,
\]

where \( \delta \) 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 \( e_j \) and \( e_k \) 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:

\[
T_{ans}(c_j \rightarrow c_k) = T_{ans}(c_k \rightarrow c_j) = \frac{e_j \cdot e_k}{||e_j|| ||e_k||}.
\]

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
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).

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

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. Under the two categories, there are 20 and 25 leaf categories, respectively. 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. 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 requiring 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.
CHAPTER 5. CATEGORY-SENSITIVE QUESTION ROUTING

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 \( q_r \):

\[
P(q_r|u_i) = \sum_{\text{Cluster}} \prod_{\omega \in q_r} P(\omega|\theta_{\text{Cluster}})^{n(\omega,q_r)} \text{con(Cluster, } u),
\]

(5.9)

In the above equation,

\[
P(\omega|\theta_{\text{Cluster}}) = (1 - \lambda)P(\omega|\text{Cluster}) + \lambda P(\omega|\text{Coll}),
\]

(5.10)

\[
\text{con(Cluster, } u) = \sum_{qa} \text{con(qa, } u),
\]

(5.11)

\[
\text{con(qa, } u) = \frac{\prod_{\omega \in q} P(\omega|\theta_{a_{u_i}})}{\sum_{(qa)'} \prod_{\omega \in q'} P(\omega|\theta_{a_{u_i}})}
\]

(5.12)

\[
P(\omega|\theta_{a_{u}}) = (1 - \lambda)P(\omega|a_{u}) + \lambda P(\omega|\text{Coll}),
\]

(5.13)

where \( \omega \) represents a term, \( n(\omega, q_r) \) means the number of times that \( \omega \) appears in \( q_r \), \( \theta_{\text{Cluster}} \) denotes the language model of the cluster, and \( P(\omega|\theta_{\text{Cluster}}) \) denotes the maximum likelihood estimation of \( \omega \) 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, \( \text{con(qa, } u) \) 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 \( \text{Coll} \) 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 $\theta_{u_i}$ denotes the language model of $u_i$’s profile, $\delta$ 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 al. [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

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

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).

<table>
<thead>
<tr>
<th>From</th>
<th>Software</th>
<th>Printers</th>
<th>Comedy</th>
<th>Lyrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming &amp; Design</td>
<td>0.2975</td>
<td>0.0251</td>
<td>0.0026</td>
<td>0.0026</td>
</tr>
<tr>
<td>Scanners</td>
<td>0.1158</td>
<td>0.5604</td>
<td>0.0014</td>
<td>0.0008</td>
</tr>
<tr>
<td>Drama</td>
<td>0.0053</td>
<td>0.0006</td>
<td>0.2593</td>
<td>0.0137</td>
</tr>
<tr>
<td>Other - Music</td>
<td>0.0102</td>
<td>0.0019</td>
<td>0.0273</td>
<td>0.1683</td>
</tr>
</tbody>
</table>

$q_r$ in QR:

$$\text{Mean QR Time}$$

- $MAP = \frac{\sum_{q_r \in Q_r} AvgP(q_r)}{|Q_r|},$ \hspace{1cm} (5.19)
- $AvgP(q_r) = \frac{\sum_{k=1}^{N_r} (P_r(k) \cdot IsAns(k))}{NRA_r},$ \hspace{1cm} (5.20)
- $P_r(k) = \frac{NRA_r(k)}{k},$ \hspace{1cm} (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).

<table>
<thead>
<tr>
<th>From</th>
<th>To</th>
<th>Software</th>
<th>Printers</th>
<th>Comedy</th>
<th>Lyrics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programming &amp; Design</td>
<td>Software</td>
<td>0.2250</td>
<td>0.0236</td>
<td>0.0116</td>
<td>0.0116</td>
</tr>
<tr>
<td>Scanners</td>
<td>Printers</td>
<td>0.1676</td>
<td>0.2671</td>
<td>0.0049</td>
<td>0.0034</td>
</tr>
<tr>
<td>Drama</td>
<td>Comedy</td>
<td>0.0136</td>
<td>0.0020</td>
<td>0.5481</td>
<td>0.0376</td>
</tr>
<tr>
<td>Other - Music</td>
<td>Software</td>
<td>0.0443</td>
<td>0.0070</td>
<td>0.0748</td>
<td>0.2922</td>
</tr>
</tbody>
</table>

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* → *Software*) are more transferable than those under different ones (e.g., *Programming & Design* → *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* → *Comedy*, *Other - Music* → *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, \(\theta\) of each answerer and \(\phi\) 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/
CHAPTER 5. CATEGORY-SENSITIVE QUESTION ROUTING

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

<table>
<thead>
<tr>
<th>Type</th>
<th>Average number of potential answerers</th>
<th>Loss of recall</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Before filtering</td>
<td>After filtering (↓% )</td>
</tr>
<tr>
<td>Severe</td>
<td>243,167</td>
<td>19,235 (↓92.09%)</td>
</tr>
<tr>
<td>Lenient</td>
<td>243,167</td>
<td>137,171 (↓43.59%)</td>
</tr>
</tbody>
</table>

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 K’s using GT-A (best results are shown in bold).

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

5.4.2 Category-sensitive Language Models

Table 5.5 and Table 5.6 report Prec@K for all algorithms with different K’s 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. 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 K’s, 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 K’s, 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 $K$'s using GT-BA (best results are shown in bold).

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

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.
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).

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>QLLM</td>
<td>0.1460</td>
<td>0.1070</td>
</tr>
<tr>
<td>BCS-QLLM</td>
<td>0.1893 (↑29.66%)</td>
<td>0.1424 (↑33.08%)</td>
</tr>
<tr>
<td>TCS-QLLM</td>
<td>0.1965 (↑34.59%)</td>
<td>0.1469 (↑37.29%)</td>
</tr>
<tr>
<td>LDALM</td>
<td>0.1695 (↑16.10%)</td>
<td>0.1281 (↑19.72%)</td>
</tr>
<tr>
<td>CBLM</td>
<td>0.0031</td>
<td>0.0024</td>
</tr>
</tbody>
</table>

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 $\delta$ 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 stop-words 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
CHAPTER 5. CATEGORY-SENSITIVE QUESTION ROUTING

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

<table>
<thead>
<tr>
<th>Method</th>
<th>MRR</th>
<th>MAP</th>
</tr>
</thead>
<tbody>
<tr>
<td>QLLM</td>
<td>0.0976</td>
<td>0.0976</td>
</tr>
<tr>
<td>BCS-QLLM</td>
<td>0.1178 (↑20.70%)</td>
<td>0.1178 (↑20.70%)</td>
</tr>
<tr>
<td>TCS-QLLM</td>
<td><strong>0.1219 (↑24.90%)</strong></td>
<td><strong>0.1219 (↑24.90%)</strong></td>
</tr>
<tr>
<td>LDALM</td>
<td>0.1133</td>
<td>0.1133</td>
</tr>
<tr>
<td>CBLM</td>
<td>0.0015 (↑16.09%)</td>
<td>0.0015 (↑16.09%)</td>
</tr>
</tbody>
</table>

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 K’s. 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 pre-computed) 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
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 $P_{\text{Uniform}}(u)$ to using $P_{\text{Indegree}}(u)$ 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.
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
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.
CHAPTER 5. CATEGORY-SENSITIVE QUESTION ROUTING

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. What's 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 an affordable hotel in Edinburgh City Center?

The above problem calls for a new approach in structurizing 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 structurize 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 structurizing questions in CQA services.

In this chapter, we propose a novel hierarchical entity-based approach (i.e., “cluster entity tree” or CET) to structurize questions in CQA services by leveraging an existing large-scale entity repository. Figure 6.1 shows how CET structurizes 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*.

“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). 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 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.
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:

$$Entropy(e_i) = - \sum_{c=1}^{\lvert C \rvert} P_c(e_i) \cdot \log(P_c(e_i)),$$

where $\lvert C \rvert$ 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 $\alpha$ and remove all candidate entities with entropy larger than $\alpha$. The setting of $\alpha$ is a tradeoff: higher values will introduce more noise, while smaller values will lead to decreased recall. In our experiments, we empirically set $\alpha$ 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\textsuperscript{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

\textsuperscript{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 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 $\theta_{max}$;
- **AC-MIN:** the similarity between entity $e_i$ and any entity $e_j$ in one of the clusters is larger than threshold $\theta_{min}$;
- **AC-AVG:** the mean similarity between entity $e_i$ and any entity $e_j$ in one of the clusters is larger than threshold $\theta_{avg}$.

---

**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: 
4: 
5: 
6: 
7: 
8: 
9: 
10: Create a new cluster with $e_i$ being the element.
11: end if
12: end while

---

6We 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]:

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

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:

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

**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.

<table>
<thead>
<tr>
<th>Yahoo! Answers</th>
<th>FreeBase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars &amp; Transportation</td>
<td>Aviation, Transportation, Boats</td>
</tr>
<tr>
<td></td>
<td>Spaceflight, Automotive, Bicycles, Rail</td>
</tr>
<tr>
<td>Computers &amp; Internet</td>
<td>Computer, Internet</td>
</tr>
<tr>
<td>Sports</td>
<td>Soccer, Olympics, Sports, American football,</td>
</tr>
<tr>
<td></td>
<td>Baseball, Basketball, Ice Hockey, Martial Arts,</td>
</tr>
<tr>
<td></td>
<td>Cricket, Tennis, Boxing, Skiing</td>
</tr>
<tr>
<td>Travel</td>
<td>Travel, Location, Transportation</td>
</tr>
</tbody>
</table>

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:

\[
\text{sim}_{DS}(t_a, t_b) = \frac{\sum_i \min(x_i, y_i)}{\sum_i x_i + \sum_i y_i - \sum_i \min(x_i, y_i)}. \tag{6.4}
\]

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.

<table>
<thead>
<tr>
<th>Category</th>
<th>Number of Questions</th>
<th>Number of Entities</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cars &amp; Transportation</td>
<td>1,220,427</td>
<td>3,267,596</td>
</tr>
<tr>
<td>Computers &amp; Internet</td>
<td>2,912,280</td>
<td>7,324,655</td>
</tr>
<tr>
<td>Sports</td>
<td>2,363,758</td>
<td>6,230,868</td>
</tr>
<tr>
<td>Travel</td>
<td>1,347,801</td>
<td>3,728,286</td>
</tr>
</tbody>
</table>

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

Set EC. This set is constructed to automatically evaluate hierarchi-
cal 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 cluster-
ing). 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 clus-
tering on Set EC is partial clustering. However, it leverages Free-
base labels and avoids manual labeling, which is time-consuming.
Furthermore, partial clustering results are enough for evaluating dif-
ferent strategies’ performance and choosing the best strategy.

Following the common practice, we evaluate entity extraction us-
ing 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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F₁</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stanford NER</td>
<td>0.750</td>
<td>0.155</td>
<td>0.257</td>
</tr>
<tr>
<td>FIGER</td>
<td>0.763</td>
<td>0.154</td>
<td>0.256</td>
</tr>
<tr>
<td>Freebase</td>
<td>0.644</td>
<td>0.595</td>
<td>0.619</td>
</tr>
<tr>
<td>Ours</td>
<td>0.647</td>
<td>0.809</td>
<td>0.719</td>
</tr>
</tbody>
</table>

**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 F₁ 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 F₁ 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 F₁ of Transportation is much higher than
Table 6.5: Clustering results of Cars & Transportation and Computer & Internet using AC-MAX ($\theta_{\text{max}}=0.1$).

<table>
<thead>
<tr>
<th>Level</th>
<th>Cars &amp; Transportation</th>
<th>Computer &amp; Internet</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Count</td>
<td>P</td>
</tr>
<tr>
<td>1</td>
<td>1,281</td>
<td>0.948</td>
</tr>
<tr>
<td>2</td>
<td>1,202</td>
<td>0.989</td>
</tr>
<tr>
<td>3</td>
<td>858</td>
<td>1.000</td>
</tr>
<tr>
<td>4</td>
<td>1,776</td>
<td>1.000</td>
</tr>
<tr>
<td>5</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Total</td>
<td>5,117</td>
<td>0.984</td>
</tr>
</tbody>
</table>

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 ($\theta_{max}=0.1$).

<table>
<thead>
<tr>
<th>Level</th>
<th>Count</th>
<th>P</th>
<th>R</th>
<th>F1</th>
<th>Count</th>
<th>P</th>
<th>R</th>
<th>F1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>890</td>
<td>0.941</td>
<td>0.883</td>
<td>0.901</td>
<td>748</td>
<td>0.972</td>
<td>0.653</td>
<td>0.743</td>
</tr>
<tr>
<td>2</td>
<td>636</td>
<td>0.978</td>
<td>0.964</td>
<td>0.963</td>
<td>200</td>
<td>0.974</td>
<td>0.730</td>
<td>0.798</td>
</tr>
<tr>
<td>3</td>
<td>492</td>
<td>0.965</td>
<td>0.882</td>
<td>0.899</td>
<td>120</td>
<td>1.000</td>
<td>0.833</td>
<td>0.890</td>
</tr>
<tr>
<td>4</td>
<td>1,080</td>
<td>0.978</td>
<td>0.844</td>
<td>0.881</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>5</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Total</td>
<td>3,098</td>
<td>0.965</td>
<td>0.886</td>
<td>0.907</td>
<td>1,068</td>
<td>0.976</td>
<td>0.688</td>
<td>0.770</td>
</tr>
</tbody>
</table>

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).

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

<table>
<thead>
<tr>
<th></th>
<th>Knowledge-learning Tasks</th>
<th>Question-search Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CET-based</td>
<td>List-based</td>
</tr>
<tr>
<td>Familiarity</td>
<td>3.18</td>
<td>3.22</td>
</tr>
<tr>
<td>Easiness</td>
<td>3.64</td>
<td>3.66</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>3.70</td>
<td>2.94</td>
</tr>
<tr>
<td>Enough Time</td>
<td>3.87</td>
<td>3.83</td>
</tr>
<tr>
<td>Helpfulness</td>
<td>4.16</td>
<td>3.03</td>
</tr>
</tbody>
</table>

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-
Similarly, 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 \( \text{cet}_e \) whose root entity is \( e \) (Line 7). Line 8-16 iteratively put questions in corresponding clusters of \( \text{cet}_e \). 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 \( \text{cet}_e \) (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 \( Q_q = \{ q_1, q_2, \ldots, q_k \} \) to \( q \), an entity repository \( ER \), an empty list \( \Theta \).

**Output:** A new ranked list of questions.

1: Extract entities from each question of \( Q \) and construct an entity co-occurrence graph;
2: Get the PageRank score of each entity;
3: if there is no entity \( e \) in \( q \) then
4: return \( Q_q \);
5: else
6: Identify the key entity \( e \) from \( q \) which has the highest PageRank score;
7: Construct the CET \( cet_e \) from \( Q \) based on \( ER \);
8: for each question \( q_i \) do
9: For all entities in \( q_i \), build an entity chain \( C \) in descending order of PageRank scores;
10: From \( C \) extract the first entity \( \hat{e} \) that is not similar to \( e \);
11: if \( \hat{e} \) exists then
12: Put \( q_i \) in the corresponding cluster of nodes;
13: else
14: Put \( q_i \) in \( \Theta \);
15: end if
16: end for
17: end if
18: Rank all clusters on \( cet_e \) according to their first elements’ original ranking;
19: Output the final ranking cluster by cluster and append the questions in \( \Theta \) at the last.

---

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

<table>
<thead>
<tr>
<th></th>
<th>VSM</th>
<th>Re-ranking</th>
<th>QLLM</th>
<th>Re-ranking</th>
</tr>
</thead>
<tbody>
<tr>
<td>MRR</td>
<td>0.3838</td>
<td><strong>0.4195</strong>*(9.30%)</td>
<td>0.3593</td>
<td><strong>0.3889</strong>*(8.24%)</td>
</tr>
<tr>
<td>MAP</td>
<td>0.3376</td>
<td><strong>0.3558</strong>*(5.39%)</td>
<td>0.3326</td>
<td><strong>0.3479</strong>*(4.60%)</td>
</tr>
<tr>
<td>Prec@1</td>
<td>0.2500</td>
<td><strong>0.3125</strong>*(25.00%)</td>
<td>0.2438</td>
<td><strong>0.2688</strong>*(10.25%)</td>
</tr>
</tbody>
</table>
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
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-
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 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 Zhi-dao 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\textsuperscript{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.

\textsuperscript{1}http://www.google.com/insidesearch/features/search/knowledge.html

\hspace{1cm} \textbf{End of chapter.}
Bibliography


