Improving Search and Recommendation by Asking Clarifying Questions

Jie Zou

Improving Search and Recommendation by Asking Clarifying Questions

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof. dr. ir. K.I.J. Maex
ten overstaan van een door het College voor Promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op maandag 6 september 2021, te 14:00 uur

door

Jie Zou

geboren te Hunan

Promotiecommissie

Promotor:		
	Prof. dr. E. Kanoulas	Universiteit van Amsterdam
Co-promotor:		
	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Overige leden:		
	Prof. dr. S. Ben Allouch	Universiteit van Amsterdam
	Prof. dr. P.T. Groth	Universiteit van Amsterdam
	Prof. dr. Z. Ren	Shandong University
	Dr. R. Fernandez Rovira	Universiteit van Amsterdam
	Dr. H. Zamani	University of Massachusetts Amherst

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

The research was supported by the China Scholarship Council.

Cover by Jie Zou
Printed by Off Page, Amsterdam

ISBN: 978-94-93197-73-2

Acknowledgment

I joined the Information Retrieval Lab (IRLab) at the University of Amsterdam to pursue my PhD in October 2017, full of excitement. Although I was optimistic at the beginning, it has still been a tough journey for the past four years. Finally, I have finished my PhD research and have the chance to defend my PhD thesis. I really appreciate the help from my supervisors, friends, and family on the road towards the PhD degree.

Firstly, I would like to sincerely thank my promotor and daily supervisor Prof. dr. Evangelos Kanoulas. He opened the door to the world of Information Retrieval for me. Thanks for giving me the opportunity to become an Information Retrieval researcher. We had many discussions about research and Evangelos always provided me with insightful comments. Thanks for your help to make all my work better. I am really impressed by your brilliance in research, high efficiency in work, and kind attitude towards your students. Evangelos, thank you for helping me to grow and teaching me how to do research.

I also want to thank my co-promotor Prof. dr. Maarten de Rijke. Thanks a lot for taking care of me and bringing me into the IRLab. Thank you for assessing my regular UvA evaluations and providing me with valuable suggestions. Also, thanks for your leadership in uniting us together, making such a big amazing group, and providing me with a super research environment. I am also honored to have Prof. Somaya Ben Allouch, Dr. Raquel Fernández, Prof. Paul Groth, Prof. Zhaochun Ren and Dr. Hamed Zamani as my committee members, and thanks a lot for your efforts to review my thesis.

I thank my colleagues at the IRLab. It was a great pleasure to work with all of you. Many thanks to Alexey, Ali A, Ali V, Amir, Ana, Andreas, Anna, Antonios, Arezoo, Artem, Barrie, Bob, Chang, Christof, Christophe, Chuan, Dan, Dat, David, Gabriel, Georgios, Hamid, Harrie, Hinda, Hongyu, Hosein, Ilya, Jiahuan, Jin, Julia, Julien, Justine, Katya, Ke, Maarten M, Maartje, Mahsa, Maria, Mariya, Marlies, Marzieh, Maurits, Ming, Mohammad, Mostafa, Mozhdeh, Nikos, Olivier, Peilei, Pengjie, Pooya, Praveen, Rolf, Ruben, Sam, Sami, Sebastian, Shangsong, Shaojie, Shubha, Spyretta, Stefan, Svitlana, Tom, Trond, Vera, Wanyu, Weijia, Xiangsheng, Xiaohui, Xinyi, Yang, Yangjun, Yifan, Zhaochun, and Ziming. I appreciate the interesting conversations and academic discussions. I also thank Petra for helping me deal with pre-defense procedures and other UvA paperwork. Also, I want to thank the friends I have met in Amsterdam: Biwen, Chenxi, Chenhui, Chunfang, Fuzhen, Huan, Jiafang, JiaoJiao, Lichun, Qiong, Shihan, Shuai, Shuo, Songyu, Xiaomeng, Xue, Yang, Yansong, Yong, Yueyao, Zhe, and special thanks to Weikang. Besides, I also want to thank Maartje for your efforts to translate my thesis summary to Dutch. Jin, Dan, thank you for being my paranymphs. I also thank the China Scholarship Council for their sponsorship.

Last but not least, I want to thank my family. Dear mom, dad, and brother, thanks for your love and support!

Jie Zou
April, 2021
Amsterdam

Contents

1 Introduction **1**
- 1.1 Research Outline and Questions . . . 2
- 1.2 Main Contributions . . . 4
  - 1.2.1 Algorithmic Contributions . . . 4
  - 1.2.2 Empirical Contributions . . . 4
  - 1.2.3 Resources . . . 6
- 1.3 Thesis Overview . . . 6
- 1.4 Origins . . . 8

I Modelling **11**

2 Question-based Document Search **13**
- 2.1 Introduction . . . 13
- 2.2 Related work . . . 16
  - 2.2.1 Technology Assisted Reviews . . . 16
  - 2.2.2 Interactive Search . . . 19
- 2.3 Methodology . . . 21
  - 2.3.1 Question Pool Construction . . . 22
  - 2.3.2 Sequential Bayesian Search for TAR . . . 23
  - 2.3.3 Accounting for Noisy Answers . . . 25
  - 2.3.4 When to Stop Asking Questions . . . 25
- 2.4 Experiments and Analysis . . . 28
  - 2.4.1 Experimental Setup . . . 28
  - 2.4.2 **RQ1.1** The Effect of the CAL Stopping Point and the Number of Questions . . . 33
  - 2.4.3 **RQ1.2** The Performance of the SBSTAR Method . . . 34
  - 2.4.4 **RQ1.3** The Effect of Noisy Answers . . . 35
  - 2.4.5 **RQ1.4** The Effectiveness of Stopping to Ask Question . . . 37
  - 2.4.6 Online User Study . . . 41
- 2.5 Conclusions and Discussion . . . 44

3 Question-based Product Search **47**
- 3.1 Introduction . . . 47
- 3.2 Related Work . . . 48
  - 3.2.1 Product Search . . . 48
  - 3.2.2 Interactive Search . . . 49
  - 3.2.3 Learning to Ask . . . 50
- 3.3 Methodology . . . 51
  - 3.3.1 Question Pool Construction . . . 51
  - 3.3.2 Cross-user Duet Learning . . . 51
  - 3.3.3 QSBPS Algorithm . . . 54
- 3.4 Experiments and Analysis . . . 56
  - 3.4.1 Experimental Setup . . . 56

3.4.2 **RQ2.1** The Impact of the Number of Questions and the Question Reward Parameter γ . . . 59
3.4.3 **RQ2.2** The Influence of Using User Reviews Data . . . 59
3.4.4 **RQ2.3** The Influence of the Duet Training . . . 59
3.4.5 **RQ2.4** The Influence of Noisy Answers . . . 62
3.4.6 **RQ2.5** The Effectiveness of QSBPS Compared with Other Algorithms . . . 64
3.5 Conclusions and Discussion . . . 66

4 Question-based Recommendation **69**
4.1 Introduction . . . 69
4.2 Related Work . . . 71
4.3 Methodology . . . 73
4.3.1 Latent Factor Recommendation . . . 73
4.3.2 Question Learning . . . 76
4.3.3 Question Asking . . . 77
4.3.4 Question-based Recommender System . . . 78
4.4 Experiments and Analysis . . . 79
4.4.1 Experimental Setup . . . 79
4.4.2 **RQ3.1** Impact of Parameters . . . 81
4.4.3 **RQ3.2** Performance Comparison . . . 83
4.4.4 **RQ3.3** Cold-start Performance Analysis . . . 84
4.4.5 **RQ3.4** Contribution of Offline Initialization . . . 86
4.4.6 **RQ3.5** Online User Study . . . 86
4.5 Conclusions and Discussion . . . 87

II Evaluation **89**

5 A User Study on Question-based Product Search **91**
5.1 Introduction . . . 91
5.2 Study Design . . . 92
5.3 Experiments and Analysis . . . 94
5.3.1 Research Questions . . . 94
5.3.2 Participants . . . 94
5.3.3 **RQ4.1** User Willingness to Answer Questions . . . 94
5.3.4 **RQ4.2** User Answers Noise . . . 95
5.3.5 **RQ4.3** User Perceived Helpfulness . . . 98
5.4 Conclusion and Discussion . . . 98

6 A User Study on Question-based Web Search **101**
6.1 Introduction . . . 101
6.2 Related Work . . . 103
6.3 Method . . . 105
6.3.1 Protocol . . . 105
6.3.2 Study Design . . . 105

6.3.3 Participants 108
6.4 Results 109
6.4.1 **RQ5.1** Impact of CQs on Search Behavior and Satisfaction . . 109
6.4.2 **RQ5.2** Impact of User Background on CQs Interactions . . . 114
6.4.3 **RQ5.3** User Engagement of CQs under Various Circumstances 116
6.5 Conclusions and Discussion 118

7 Conclusions **121**
7.1 Main Findings 121
7.2 Limitations and Future Directions 123
7.2.1 Question-based Document Search 123
7.2.2 Question-based Product Search 124
7.2.3 Question-based Recommendation 124
7.2.4 A User Study on Question-based Product Search 125
7.2.5 A User Study on Question-based Web Search 125
7.2.6 Clarifying Questions 125

Bibliography **127**

Appendices **139**
A Instructions for the User Study in Chapter 5 139
B Instructions for the User Study in Chapter 6 140

Summary **143**

Samenvatting **145**

1 Introduction

Search and recommender systems are prevalent in our daily lives. They help us to deal with an increasingly complex information landscape [136]. Search systems enable users to quickly and easily access information relevant to what they are looking for by submitting queries. Recommender systems try to predict user preferences so as to recommend particular items to satisfy their wishes and needs [88]. There is a subtle difference between the two types of systems. A search system typically acts in a passive manner where the search is initiated by the user query expressing a more or less clear user intent, while a recommender system acts more proactively. At the same time they share many components, and as output they both offer users a ranked list of items [63].

Traditional search and recommendation methods measure the relevance of a document to a query or the preference of a user for an item, by training a matching function. While these matching functions work to some extent, their performance can still be limited by the semantic gap between a document and a query or between a user and an item [200]. Compared to traditional approaches, interactive search and recommender systems place the *user in the loop* so as to collect feedback from users to generate more accurate search and recommendation results. Existing methods query users for feedback in various ways. They may directly elicit user preferences over items/documents either implicitly or explicitly [31, 35, 71], or collect and learn from human-to-human conversations [27, 113, 170], or ask clarifying questions (CQs) on the basis of some "anchor" text (e.g., item aspects [213], entities [219, 224, 225], or grounding text [143]) that characterizes the items/documents. In this thesis, we focus on the latter one: asking CQs.

Search and recommender systems that take the initiative to ask CQs to better understand users' information needs are receiving increasing attention from the research community. In search systems, the search queries formulated by users are often short, ambiguous, incomplete, or faceted, and therefore too general to capture the minute details of the items/documents that a user is looking for. In recommender systems, user preferences and hence recommendations are generated based on users' explicit information (e.g., a user profile [86]) and implicit historical behavior [18]. However, user preferences are often dynamic and might evolve over time due to internal or external factors. Moreover, users' information and historical behavior are unknown in cold-start settings (i.e., when users are new to the system, and therefore the system has no data associated with them), especially when the increasing concern of privacy

is considered. Therefore, researchers have augmented search and recommendation functionality by allowing systems to ask CQs, with users enabled to converse with the system, to better understand users' information needs [5, 206] and better control recommendation loops [146].

The use of CQs is a relatively new and useful technique to help search and recommender systems better recognize users' intent, context, and preferences. While building systems capable of having mixed-initiative interactions with users has been a long-standing goal [15, 60, 144], we have observed notable developments and successes in this area only recently [5, 6, 79, 191, 206, 219, 225]. The significance and effectiveness of CQs has been proved in a broad range of use cases such as product search [213], information-seeking conversations [5, 79, 104, 191, 206, 207], and dialogue systems [49, 168]. These recent publications showcase the effectiveness of CQs for different applications, yet research on the algorithms using CQs to improve search and recommender systems is still limited, compared to traditional search and recommender systems. Moreover, these recent publications mainly focus on the system performance, yet the impact of asking CQs on users is unclear. There is no empirical study to quantify whether and to what extent users are willing or able to answer these CQs, while understanding the extent of the impact of CQs on user's behavior and their ability to identify relevant information also remains relatively unexplored.

In this thesis, we investigate how to effectively and efficiently ask CQs to the users, in order to support search and recommendation. Specifically, we first explore the automatic construction and algorithmic selection of CQs to improve search systems in tasks that range from locating scientific documents to products. Then, we explore the use of CQs to improve recommender systems in order to recommend the best item to the users. Last, we conduct a series of online user studies to understand users' interactions with CQs in search and recommender systems.

1.1 Research Outline and Questions

In this thesis, we attempt to answer the following five research questions, with the first three focusing on generating and using CQs to optimize search and recommender systems and the last two focusing on understanding user interactions with CQs, by conducting user studies. Below, we detail our research questions.

RQ1 How can we ask CQs to effectively retrieve documents?

We first explore how we can ask CQs to effectively retrieve documents. We test it in the domain of systematic reviews for locating scientific documents, where total recall is necessary. Total recall, i.e., finding all relevant documents, is a harder problem than regular document search, and Continuous Active Learning (CAL) methods have proven effective in finding most of the relevant documents in a collection [135] but fail to find the last few relevant documents. We find that their performance is reaching a plateau when 80%–90% of the relevant documents have been found. Finding the last few relevant documents typically requires exhaustively reviewing the whole collection, which leads to high effort and high cost. This motivates us to study locating the last few relevant documents by

asking CQs. To answer **RQ1**, we propose a novel interactive algorithm to aid document search, which can efficiently retrieve these last few, but significant, relevant documents by asking CQs. We also devise a sequential Bayesian search method that selects the optimal sequence of questions to ask to the user.

RQ2 How can we effectively ask CQs to improve product search?

Product search is generally recognized as the first and foremost stage of online shopping and thus significant for users and retailers of e-commerce. Most of the traditional retrieval methods use some similarity functions to match the user's query and the document that describes a product, either directly or in a latent vector space, which usually cannot capture the minute details of the user desired items [200]. This motivates us to study product search by asking CQs. To answer **RQ2**, we propose a novel interactive product search algorithm, the Question-based Sequential Bayesian Product Search method (QSBPS), to effectively locate the best matching product by asking CQs. The method uses duet training, which learns the product relevance as well as the reward of the potential CQs to be asked to the user by using historical data.

RQ3 How can we effectively ask CQs to improve recommender system performance?

Compared to traditional recommender systems, research in the field of conversational and question-based recommender systems is rather limited. Previous conversational recommender systems ask users to express their preferences over items or item facets. Instead, we ask users to express their preferences over descriptive item features by asking CQs. To answer **RQ3**, we propose a novel interactive algorithm, Question-based recommendation (Qrec), to assist users to find items interactively, by answering automatically constructed and algorithmically chosen questions.

RQ4 To what extent can users answer CQs of question-based product search systems?

While existing publications have demonstrated success of asking CQs in helping systems to better understand users, most of them evaluate algorithms on whether the system can place the target item at a high ranking position assuming that users can perfectly answer these questions. To the best of our knowledge, there is no empirical validation of whether and to what extent users can respond to these questions, and the usefulness perceived by users while interacting with the system. To answer **RQ4**, we conduct a user study by deploying an online CQ-based product search system. We collect both implicit interaction behavior data and explicit feedback from users to explore to what extent users are willing to answer CQs and are able to correctly answer CQs.

RQ5 How do users interact with CQs in web search?

Although recent publications have demonstrated the effectiveness of CQs on system performance and users enjoy query clarification, the impact of the quality of CQs on user's behavior and their ability to identify relevant information remain unstudied. In **RQ5**, we conduct a large user study in order to understand how users interact with different quality categories of CQs under different circumstances

and how their search performance in terms of finding relevant information, their search behavior and their satisfaction are affected by the quality of the CQs.

1.2 Main Contributions

In this section, we summarize the main contributions of this thesis.

1.2.1 Algorithmic Contributions

1. Question-based Document Search (Chapter 2)
   (1) A question construction method to construct a set of questions to be asked to users in terms of entities contained in the documents of the collection.
   (2) A novel question-based retrieval method for locating the last few relevant documents, SBSTAR.
   (3) An extension, $\text{SBSTAR}_{\text{ext}}$, to automatically decide when to stop asking questions.
   (4) Two question selection methods to select the optimal sequence of questions to ask, one version without user answer noise and one version accounting for the user's erroneous answers.
   (5) Three rudimentary simulation methods of users' noisy answers when answering the generated questions.
2. Question-based Product Search (Chapter 3)
   (1) A novel question-based product search method for locating the best matching product based on constructed questions, QSBPS.
   (2) A method that learns question reward and cross-user system belief with limited data.
3. Question-based Recommendation (Chapter 4)
   (1) A novel question-based recommendation method, Qrec.
   (2) A novel algorithm framework, that incorporates online matrix factorization and online users' belief tracking for sequential question asking.
   (3) A novel matrix factorization method which can incorporate the offline training and efficient online updating of the user and item latent factors.

1.2.2 Empirical Contributions

4. Question-based Document Search (Chapter 2)
   (1) An empirical comparison of the proposed model SBSTAR, including a no-user-answer-noise version and a user-answer-noise-tolerance version, with other state-of-the-art baselines on both abstract-level relevance and document-level relevance.

(2) An empirical comparison of different classification models on when to stop asking questions.

(3) An analysis of the importance of different features on when to stop asking questions.

(4) An analysis on the basis of user simulations on the noise tolerance of the proposed algorithm SBSTAR.

(5) An analysis of the impact of parameters, the stopping point and the number of questions asked, on the model performance.

(6) A small user study to validate the assumptions made regarding the users' willingness to answer a number of questions, their efforts, and their noisy answers.

5. Question-based Product Search (Chapter 3)

(1) An empirical comparison of the proposed product search model QSBPS with other state-of-the-art product search baselines.

(2) An analysis of the impact of parameters, the number of questions asked and the question reward trade-off parameter, on the model performance.

(3) An extensive analysis of different modules of the proposed product search algorithm QSBPS, including an analysis of duet training, user reviews data and noise tolerance in user answers.

6. Question-based Recommendation (Chapter 4)

(1) An empirical comparison of the proposed recommendation model Qrec with other state-of-the-art recommendation baselines.

(2) An analysis of the impact of parameters, the online updating trade-off parameter, the dimension of the latent factors, and the number of questions asked, over the effectiveness of our model.

(3) An extensive analysis of the model performance for cold-start users and cold-start items.

(4) An empirical analysis of the offline initialization of the model.

(5) A small online user study to validate the the effectiveness of our method.

7. A User Study on Question-based Product Search (Chapter 5)

(1) A user study to validate to what extent users willing to engage with a question-based product search system.

(2) A user study to examine to what extent users provide correct answers to the generated questions

(3) A user study to examine how useful users perceive a question-based product search system to be while interacting with it.

8. A User Study on Question-based Web Search (Chapter 6)

(1) A large user study to understand to what extent asking CQs affects users' search behavior and satisfaction on Web search.

(2) A large user study to analyze how much user background and task perception affect their interactions with CQs on Web search.

(3) A large user study to understand how users interact with CQs under various circumstances on Web search.

1.2.3 Resources

9. An open source implementation of SBSTAR and SBSTAR$_{\text{ext}}$ (Chapter 2).

10. An open source implementation of QSBPS (Chapter 3).

11. An open source implementation of Qrec (Chapter 4).

1.3 Thesis Overview

In this section we present an overview of the thesis.

Chapter 2: Question-based Document Search. In this chapter, we first propose a novel model to efficiently retrieve documents by asking CQs. Specifically, we instantiate document search in the domain of systematic reviews for locating scientific documents, where the majority of relevant documents are found but identifying the last few missing relevant documents is needed for achieving the requirement of total recall. The model is based on constructing questions about the presence or absence of entities in the missing relevant documents. The hypothesis made is that entities play a central role in documents carrying key information, and that the users are able to answer questions about the presence or absence of an entity in the missing relevance documents. Based on this we devise a sequential Bayesian search method that selects the optimal sequence of questions to ask. We then extend it by (a) investigating the noise tolerance of the proposed algorithm; (b) proposing an alternative objective function to optimize, which accounts for the user's "erroneous" answers; (c) proposing a method that sequentially decides the best point to stop asking questions to the user; and (d) conducting a small user study to validate some of the assumptions made in the chapter. The experimental results demonstrate that the proposed algorithms can greatly improve performance, requiring less reviewing effort to find the last relevant documents compared to state-of-the-art methods, even in the case of noisy answers. Further, they show that our algorithm learns to stop asking questions at the right time. Last, we conduct a small user study involving an expert reviewer. The user study validates some of the assumptions made in this chapter regarding the user's willingness to answer the system questions and the extent of it, as well as the ability of the user to answer these questions.

Chapter 3: Question-based Product Search. In this chapter, we propose a novel interactive method to effectively locate the best matching product. The method is based on the assumption that there is a set of candidate questions for each product to be asked. In this chapter, we instantiate this candidate set by making the hypothesis that products can be discriminated by the entities that appear in the documents associated with them. We propose a Question-based Sequential Bayesian Product Search method, QSBPS, which directly queries users on the expected presence of entities in the relevant product documents. The method learns product relevance as well as the reward of the potential questions to be asked to the user by being trained on the search history and purchase behavior of a specific user together with that of other users. Our experimental results show that the proposed method can greatly improve the performance of product search compared to the state-of-the-art baselines.

Chapter 4: Question-based Recommendation. In this chapter, we propose a novel **Q**uestion-based **rec**ommendation method, Qrec, to assist users to find items interactively, by answering automatically constructed and algorithmically chosen questions. The model is first trained offline by a novel matrix factorization algorithm, and then iteratively updates the user and item latent factors online by a closed-form solution based on the user answers. Meanwhile, our model infers the underlying user belief and preferences over items to learn an optimal question-asking strategy by using Generalized Binary Search (GBS), so as to ask a sequence of questions to the user. Our experimental results demonstrate that our proposed matrix factorization model outperforms the traditional Probabilistic Matrix Factorization model. Further, our proposed Qrec model can greatly improve the performance of state-of-the-art baselines, and it is also effective in the case of cold-start user and item recommendations.

Chapter 5: A User Study on Question-based Product Search. In this chapter, we conduct an online experiment by deploying an experimental system, which interacts with users by asking CQs against a product repository. We collect both implicit interaction behavior data and explicit feedback from users to explore (1) user willingness to answer CQs: are users willing to answer the CQs, how many of them, when do they stop and why, and how fast do they provide answers; (2) to what extent can users provide correct answers and what factors affect this; and (3) how is the user perceived helpfulness for the CQ-based system. Some of the findings of the study contradict current assumptions on simulated evaluations in the field, while they point towards improvements in the evaluation framework and can inspire future interactive system designs.

Chapter 6: A User Study on Question-based Web Search. In this chapter, we conduct a large user study to understand how users interact with different quality categories of CQs and how their search performance in terms of finding relevant information, their search behavior and their satisfaction are affected by the quality of the CQs. Analysis from implicit interaction data and explicit user feedback reveals that high quality CQs improve user performance and satisfaction, while low and mid quality CQs are harmful, in which case allowing the users to complete their tasks without CQ support may be preferred. We also observe that user engagement, and therefore the need for CQ support,

is affected by various factors, such as search result quality or perceived task difficulty. Findings from this study can help researchers and system designers realize why, when, and how users interact with CQs, leading to a better understanding and design of search clarification systems.

Chapter 7: Conclusions. Finally, in Chapter 7, we conclude the thesis and discuss some future directions.

1.4 Origins

The main research chapters in the thesis are based on the following papers:

Chapter 2 is based on the following paper:

- J. Zou, D. Li, and E. Kanoulas. Technology assisted reviews: Finding the last few relevant documents by asking yes/no questions to reviewers. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 949–952, 2018.
- J. Zou and E. Kanoulas. Towards question-based high-recall information retrieval: Locating the last few relevant documents for technology-assisted reviews. *ACM Trans. Inf. Syst.*, 38(3), May 2020.

The latter journal paper is an extension of the former conference paper. JZ designed the model, implemented the model, ran experiments and did most of the writing. All authors contributed to the design and discussion of the model. EK contributed to the writing.

Chapter 3 is based on the following paper:

- J. Zou and E. Kanoulas. Learning to ask: Question-based sequential bayesian product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 369–378, 2019.

JZ designed the model, implemented the model, ran the experiments, and did most of the writing. EK helped with reformulating the idea and the writing.

Chapter 4 is based on the following paper:

- J. Zou, Y. Chen, and E. Kanoulas. Towards question-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 881–890, 2020.

JZ designed the model, implemented the model, performed the experiments, and did most of the writing. YC helped with the model design. EK contributed to the writing.

Chapter 5 is based on the following paper:

- J. Zou, E. Kanoulas, and Y. Liu. An empirical study on clarifying question-based systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2361–2364, 2020.

JZ was responsible for the experimental design, implemented the system, ran the experiments, and did most of the writing. EK, and YL helped with the writing.

Chapter 6 is based on the following paper:

- J. Zou, M. Aliannejadi, E. Kanoulas, M. S. Pera, and Y. Liu. Users meet clarifying questions: Towards a better understanding of user interactions for search clarification. *ACM Trans. Inf. Syst.*, 2021. Submitted.

JZ was responsible for most of the experimental design, implemented the system, ran the experiments, and did most of the writing. All authors contributed to the experimental design and result analysis. MA helped with the system implementation. MA, EK, MSP helped with the writing.

The writing of the thesis also benefited from work on the following publications:

- Y. Chen, Y. Wang, X. Zhao, J. Zou, and M. de Rijke. Block-aware item similarity models for top-n recommendation. *ACM Trans. Inf. Syst.*, 38(4), Sept. 2020.
- N. Kondylidis, J. Zou, and E. Kanoulas. Category aware explainable conversational recommendation. In *Workshop on Mixed-Initiative ConveRsatiOnal Systems 2021*, 2021.

Part I

Modelling

2 Question-based Document Search

In this chapter, we propose a novel question-based search algorithm to effectively retrieve the last few, but significant, documents by asking clarifying questions (CQs) about the expected presence of an entity, including a no-user-answer-noise version and a user-answer-noise-tolerance version. The search method is applied in the domain of systematic reviews for locating scientific documents, specifically the task called Technology-Assisted Review (TAR), where total recall is mandatory. In TAR, Continuous Active Learning (CAL) methods are used to find the majority of the relevant documents but are incapable to find the last few relevant documents. We aim to fill this gap by asking CQs to effectively retrieve these last few relevant documents, and answer the following research question:

RQ1 How can we ask CQs to effectively retrieve documents?

2.1 Introduction

TAR aims at locating all relevant documents in a collection while minimizing the manual effort required to review irrelevant documents. Successful applications of TAR include electronic discovery in legal proceedings [35, 133], systematic reviews in evidence-based medicine [135], and test collection construction in Information Retrieval (IR) evaluation [42, 159]. A significant research question in TAR is how to minimize the human effort required to review irrelevant documents while finding (nearly) all relevant documents, given that the cost of assessing a large document collection is high, especially when the assessors are domain experts or information specialists. To reduce this cost TAR is typically performed as a three-phase process: (a) a Boolean query is carefully designed by an information specialist expressing what constitutes relevant information to search for in a document corpus,[1] (b) potentially relevant documents are identified within the result set of the Boolean query, by experts examining only a summary of these documents (typically the title and the abstract), and (c) the documents to be included in the review are located by experts reading the full text of the document that corresponds to the relevant abstracts. The most expensive phase in this process is

This chapter was published as [220, 224].

[1] In some cases during the first phase a handful of relevant documents is also available; we do not consider this case in this chapter.

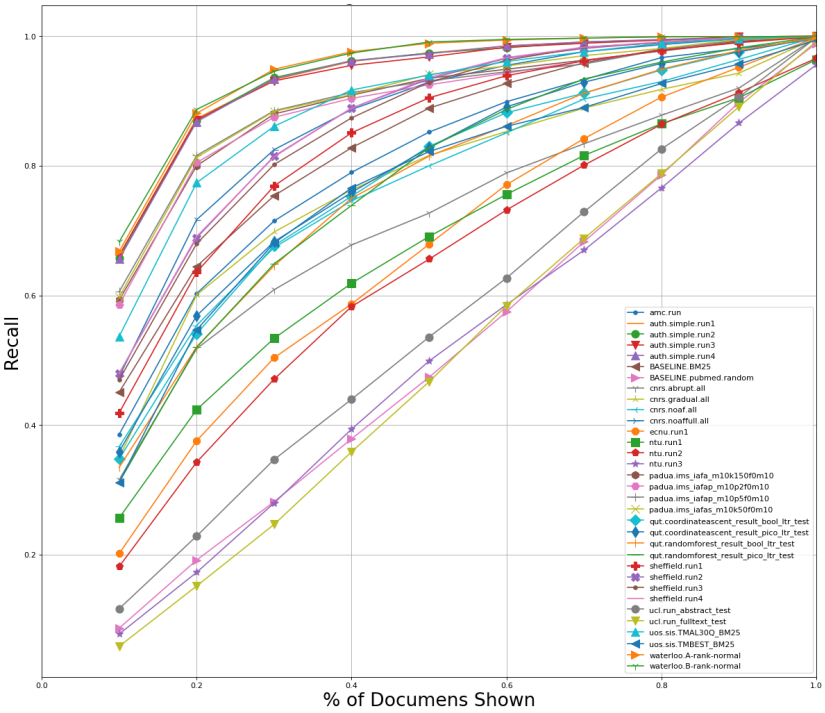
the second one, the screening of titles and abstracts, since the Boolean query typically returns a rather large dataset, in the order of thousands.

Active learning techniques, which iteratively improve the prediction accuracy by interacting with the reviewers, are considered the state-of-the-art in TAR [135]. In particular, Cormack and Grossman [35, 41] have proposed a CAL algorithm, called Baseline Model Implementation (BMI), which achieves the best performance in a number of high-recall tasks [70, 93, 224]. BMI repeatedly trains a logistic regression model to predict the relevance of documents. In every session, BMI returns the top-scored documents to users (i.e., expert reviewers) to review and label. Then, these labeled documents are added to the training dataset to re-train the logistic regression model. To speed-up the process of re-training, instead of a single document, a batch of documents is returned to the user at every iteration and the batch size increases exponentially with the iterations [35, 41]. In this way, if E is the number of labeled documents at the end of the process, the number of iterations, and hence re-training steps, is $O(\log E)$. While CAL algorithms have shown to be effective in finding relevant documents in a collection [35, 71], the percentage of relevant documents identified typically reaches a plateau at 80%–90% of all relevant documents in the collection. This often happens after reviewing and labeling 20%–40% of the collection [93, 171]. This is illustrated in Figure 2.1, which presents recall as a function of the number of documents manually reviewed in the CLEF 2017 e-Health Lab [93]. Finding the remaining 10% of relevant documents (typically 1 to 3 relevant documents) needs reviewing almost the entire collection.

To overcome the above challenge, we aim to retrieve these last few missing relevant documents by asking direct questions to reviewers about the information carried in the missing documents, instead of requesting relevance feedback on them. Our hypothesis is that asking direct questions to reviewers will allow an algorithm to discover the missing documents faster than when requesting relevance feedback on documents through CAL. Hence, we propose a Sequential Bayesian Search [193] based method for TAR, called SBSTAR. SBSTAR applies CAL up to a certain number of documents reviewed, e.g., 20%–40% of the collection. Then, it switches to directly asking yes/no questions to reviewers focusing on questions about the expected presence of an entity in the missing relevant documents. In SBSTAR, TAGME is used to identify entities, therefore an entity is defined as a sequence of informative terms (also called spots) in the input text [61]. SBSTAR works by constructing a prior belief over document relevance on the basis of the ranking model learned by CAL. Based on the prior belief, it applies Generalized Binary Search (GBS) over entities to find the optimal entity, i.e., the one that dichotomizes the probability mass of the modeled document relevance, to ask to the reviewer. After the question has been answered by the reviewer, a posterior belief is obtained to be used for the selection of the next question, or the final ranking of documents.

Since the reviewers may not always provide the correct answer to the question asked, we also propose a user-answer-noise-tolerance version of SBSTAR by incorporating the chance that user may erroneously answer the system questions, along with the no-user-answer-noise version. Further, in SBSTAR the number of questions to be asked is a predefined parameter. Setting this number depends on human experience, previous empirical results, or manual search (e.g., grid search) in a cross validation setup, which

Figure 2.1: Recall achieved by participating runs at CLEF 2017 e-Health task at different % of reviewed documents.



is costly, not flexible, and typically not optimal. Therefore, we propose the SBSTAR$_{\text{ext}}$ algorithm, which determines when to stop asking questions online and automatically, without the burden of predefining this parameter. SBSTAR$_{\text{ext}}$ continues asking questions until a stopping criterion is met. The stopping criterion is set by our trained classifier based on dynamically extracted features. Once the prediction by the trained classifier is to "stop asking," the posterior belief is used to produce the final relevant documents list. Last, we conduct a small user study to validate some of the assumptions that the reviewers are willing to answer a number of questions and the effort for answering a direct question about entities are at most as much as providing the relevance of a document.

To sum up, the main contribution of this chapter[2] is:

E1 We propose a method to construct a set of questions to be asked to the reviewers in terms of entities contained in the documents of the collection. (See Section 2.3.1.)

E2 We propose a novel interactive method, SBSTAR, that directly queries reviewers about the expected presence of an entity in relevant documents, and updates the prior belief on document relevance at every round of interaction. (See Section 2.3.2.)

[2]The source code is released in the following repository: https://github.com/JieZouIR/TAR.git

E3 We propose three rudimentary models of reviewers' noisy answers when answering the generated questions. (See Section 2.4.1.)

E4 We conduct an analysis on the basis of user simulations on the noise tolerance of the algorithm. (See Section 2.4.4.)

E5 We propose a new objective function that accounts for the user's erroneous answers when selecting the next question to ask, and demonstrate its effectiveness. (See Sections 2.3.3 and 2.4.4.)

E6 We propose $\text{SBSTAR}_{\text{ext}}$ with a novel method to decide when to stop asking questions, and demonstrate its effectiveness. (See Sections 2.3.4 and 2.4.5.)

E7 All experiments are run both against abstract-level relevance and document-level relevance, with the latter being a harder problem, since relevant documents are only a fraction of the relevant abstracts. (See Section 2.4.)

E8 We conduct a small user study to validate the assumptions made regarding the users' willingness to answer a number of questions, their efforts, and their noisy answers. (See Section 2.4.6.)

To the best of our knowledge this is the first work that attempts to ask explicit questions to reviewers for the purpose of achieving total recall that goes beyond document relevance feedback. The evaluation results show that our approach can significantly reduce human effort, while achieve high recall.

The rest of this chapter is organized as follows. In Section 2.2, we summarize related work. In Section 2.3, we introduce our approach and describe it in detail. Section 2.4 includes the experimental setup, the experimental results, and the corresponding analysis. The conclusions and discussion of this chapter are presented in Section 2.5.

2.2 Related work

2.2.1 Technology Assisted Reviews

A TAR aims at locating as many relevant documents as possible in a collection, i.e., a high recall task. The Text Retrieval Conference ("TREC") [183] has a history of studying the problem of high recall, starting with the TREC Legal track [14, 46, 84, 132, 177], followed by the TREC Total Recall track [70, 152], which received a lot of attention in 2015 and 2016. Then, CLEF [62] focuses on the total recall problem of TAR in empirical medicine [93, 171]. BMI [35], an AutoTAR CAL method, was provided to the participants of the Total Recall track as a baseline for comparison. No method evaluated in these tracks outperformed BMI, and thus BMI is recognized as the state-of-the-art [70, 152, 210]. It is a relevance-feedback method using supervised machine learning. Cormack and Grossman [35] found CAL outperforms traditional supervised learning (i.e., "simple passive learning" (SPL)) and active learning (i.e., "simple active learning" (SAL)). Yu et al. [203] also confirmed that CAL is effective for systematic reviews. Abualsaud et al. [1] designed and implemented an efficient high-recall information retrieval system using CAL. CAL [35] is an active learning method, which uses search as a first step to identify an initial set of potentially relevant and irrelevant documents for training a classifier, then presents a set of documents most likely to be relevant to the user, and solicits their feedback on their relevance. Then

Table 2.1: A running example. The topic describes the topic of the systematic review conducted. The Query reflects the Boolean query designed for this systematic review. The documents with IDs 22508578, 17567931, 16534774, 15877567, 14511167, 11113982 are relevant documents found by the CAL algorithm. In this example there is a single missing document, which is shown next. This is the document that our interactive algorithm will attempt to locate. Each candidate document, that is each document in the set of returned documents by the Boolean query, which have not been reviewed yet by the reviewer, is annotated by an entity recognition algorithm. The entities recognized in the missing document are shown in the example indicated by blue square brackets. The question pool is then constructed by using all of the entities in all the candidate documents. E.g., the entity [study] appears in the missing relevant document, while [medication] does not, but instead appears in some irrelevant candidate documents. For each question round, we select the question from the question pool to ask to the user. We do that by selecting the question with the lowest score calculated by our objective function. In our example, in the first round we selected the question "Are the documents about [study]", which received a score of 0.09 and it was the lowest score in that round, while in the second round we chose the question "Are the documents about [patient]" which had a score of 0.01 and this was the lowest score in that round.

Topic: Mini-Cog for the diagnosis of Alzheimer's disease dementia and other dementias within a community setting
Query: mini-Cog OR (MCE and (cognit* OR dement* OR screen* OR Alzheimer*))
Already found relevant documents by CAL: ID: 22508578, ID: 17567931, ID: 16534774, ID: 15877567, ID: 14511167, ID: 11113982 **Target document (missing relevant document):** ID: 20473827, Title: [Evaluation] of the [Functional Activities Questionnaire (FAQ)] in [cognitive screening] across four [American] [ethnic groups]. Abstract: The purpose of this [study] was to examine the performance of the [Functional Activities Questionnaire (FAQ)] in four [American] [ethnic groups] (N = 691), evaluate the influence of [demographic factors] and [depressive symptoms] on the [FAQ] and compare its performance with two [cognitive screening] [measures], the Mini-Cog and the [MMSE] ...
Asked questions from question pool:
Question: Are the documents about [study] (0.09) ? Answer: Yes/No/Not Sure
Question: Are the documents about [patient] (0.01)? Answer: Yes/No/Not Sure
Question: Are the documents about [cognitive screening] (0.15)? Answer: Yes/No/Not Sure
Question: Are the documents about [evaluation] (0.18)? Answer: Yes/No/Not Sure
Question: Are the documents about [dementia] (0.30)? Answer: Yes/No/Not Sure
Question: Are the documents about [medication] (0.29)? Answer: Yes/No/Not Sure
Question: Are the documents about [clinic] (0.29)? Answer: Yes/No/Not Sure
...

it uses these labeled documents as a training set to re-train the classifier. In detail, CAL first (1) creates a (set of) potentially relevant and irrelevant document(s); this can be done by different means; past work used the description of the TAR topic as a relevant document and a sample of 100 documents from the collection as the set

of irrelevant documents; then (2) trains a machine learning algorithm (e.g., a Support Vector Machine (SVM), or logistic regression) on this training set and uses it to predict the next most-likely relevant documents; typically a batch of documents is returned at every iteration with the batch size increasing exponentially with the iterations to speed-up the re-training process [35, 41]; (3) collects the relevance feedback for all of the presented documents, and amends the training set; and (4) repeats (2) and (3) until some stopping criterion is met, e.g., none of the presented documents is relevant. Despite its effectiveness, the method suffers from not being able to locate the last few relevant documents [93, 224].

Evaluation and comparative studies around TAR methods have also attracted the attention of researchers. Zhang et al. [212] used a simulation framework to evaluate sentence-level relevance feedback. Zhang et al. [211] conducted a controlled user study with 50 users to evaluate a retrieval system using the full document or selected paragraph as relevance feedback in CAL. McDonald et al. [129] presented an evaluation of active learning strategies for sensitivity reviews. The evaluation of user-in-the-loop systems has also been studied using human subjects and simulated human responses [44, 166, 167]. Grossman et al. [71] performed a comparative study on automatic and semi-automatic document selection for the TREC 2016 Total Recall track.

The summary or abstract of documents has been shown to be an effective information source for accurate and efficient relevance judgments. Tombros and Sanderson [176] found reviewers could locate more relevant documents by reviewing the extracted summary, while making fewer labeling errors. Further, Sanderson [158] found that "reviewers can judge the relevance of documents from their summary almost as accurately as if they had access to the document's full text." They showed that an assessor took 61 seconds to assess each full document while spending 24 seconds to assess each summary on average. Zhang et al. [212] also suggested that a system that presents relevant sentences could reach high recall more efficiently compared to a system that presents the entire document. Systematic reviews also use article abstracts to filter out irrelevant articles efficiently.

A number of publications that aim at deciding when to stop reviewing articles in TAR have been presented in the past. Wallace et al. [185] and Yu et al. [204] stop training their models when a pre-defined number of relevant studies is found. Di Nunzio [50] proposed a variable threshold approach to stop labeling once the percentage of non-relevant documents over the total number of judged documents reaches a fixed threshold. The most recent approach proposed, called the "knee" method [37, 38, 40, 41], uses a simple geometric criterion [161] to make a stopping decision. The "knee" method uses the fall-off in the slope of the gain curve (number of judged relevant documents vs. review effort) as a stopping criterion. However the "knee" method is a heuristic method and does not indicate how many missing relevant documents are there. Cormack and Grossman [39] proposed the SCAL method, which first estimates the number of relevant documents R in a collection by randomly sampling and labeling a large subset of the documents. Di Nunzio [50] proposed a heuristic thresholding method, the two-dimensional BM25, based on the interaction of the two probabilities used by the BM25 model: $P(d|R)$ and $P(d|NR)$ – the probability of observing document d given the currently judged relevant documents R, and the currently judged non-relevant documents NR. Their method stops judging once the proportion of non-relevant

documents over the total number of judged documents exceeds a fixed value. There are more studies that attempt to estimate the number of relevant documents, R, based on which one can decide when to stop presenting documents to the user of a search system [9, 122, 186]. Arampatzis et al. [9] proposed methods to select the cut-off point where to stop reading a ranked List that optimizes a given evaluation metric. Losada et al. [122] proposed a diversified group of stopping methods and proposed a method to estimate the number of relevant documents R. They estimate R based on a power law distribution and the similarity between the pattern of the relevance of the test query and the pattern of the relevance of each training query. All the aforementioned methods need extra assessment cost to estimate the number of relevant documents R to decide when to stop the TAR process. Our $\text{SBSTAR}_{\text{ext}}$ method differs by (1) training a classifier based on dynamic features, and (2) automatically determining when to *stop asking questions* (3) without relying on a predefined threshold or evaluation metrics.

2.2.2 Interactive Search

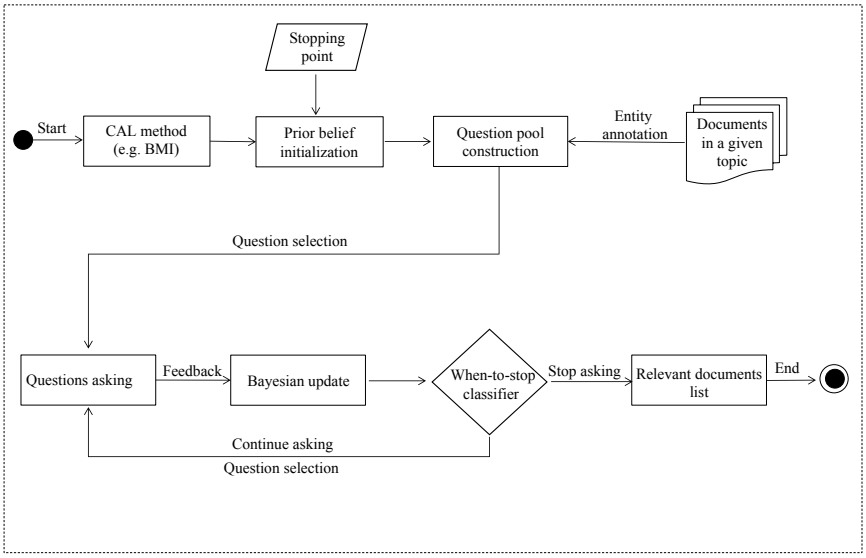
Interactive Information Retrieval has always received significant attention in the research community [22, 26, 156]. Compared with the non-interactive approaches, interactive information retrieval achieves high recall with a human-in-the-loop. Interactive approach suggests putting the human in the loop and learning a relevance model throughout an interactive search process, where users provide feedback on the relevance of presented documents, and the model adapts to this feedback [36, 69]. TREC introduced the "pooling method" [167], which selects the top-ranked documents for assessment, and recognizes all other documents as non-relevant. Then the Interactive Searching and Judging (ISJ) method has been shown to get comparable quality relevance to the pooling method with considerably less effort. ISJ repeatedly formulates queries and examines the top results of a relevance-ranking search engine [45]. Most of the methods use a special treatment of the query [109, 150, 151, 157, 209], typically expanding it with terms from labeled documents. However, query expansion has shown suboptimal performance [35], in part because handling the relationship between the original query and feedback documents is challenging [123]. Quantifying relevance on the basis of users' queries, or learning a model of relevance from past queries, cannot always capture the minute details of relevance [35, 69, 224]. Active learning [105], and multi-armed bandits [55, 85, 145] have been proposed to iteratively learn task-specific models, however, they both suffer from the multi-modality of relevance. Cormack and Mojdeh [44] proposed a combination of ISJ and CAL. Cormack and Grossman [36] proposed a continuous learning-based algorithm, BMI, for TAR, which works by iteratively training an SVM classifier on user's relevance feedback over the predicted most relevant documents. The proposed algorithm is considered state-of-the-art. However, the human effort remains extremely high for trying to find the last few relevant documents [93, 224]. Different from the aforementioned methods, which focus on receiving feedback at the level of documents, our interactive method asks explicit questions to the users in terms of entities contained in the documents of the collection.

Interactive retrieval methods have also been studied in community-based question answering (cQA). Successful applications in the field include expert finding [216, 217], question retrieval [12, 30], understanding and summarizing answers [120], question

routing in providing answers for unanswered questions [112], and inference rules discovery from text [116]. Zhao et al. [217] proposed a random-walk based learning method with recurrent neural networks from a novel viewpoint of learning ranking metric embeddings to search the right experts for answering the questions. To solve the problem of lexical gaps between questions, Chen et al. [30] presented a model to retrieve similar questions for cQA platforms to resolve users' queries by applying random walk with a recurrent neural network. They highlight a valuable investigation for considering both question contents and the asker's social interactions. Bae and Ko [12] instead presented a translation-based language model to solve the lexical gap problem for retrieving questions. To solve the cold-start problems, Wan et al. [188] and Zhao et al. [216] exploited knowledge from multiple sources to support question answering. A hybrid system to retrieve expertise to help to answer questions has also been proposed [106]. Liu et al. [120] suggested that users can reuse the best answers from similar questions as search result snippets, and highlighted the effectiveness of applying automatic summarization techniques to summarize answers. Li and King [112] introduced the concept of Question Routing to retrieve suitable questions to the right answerers to answer. They proposed a Question Routing framework by considering not only users' expertise but also the availability of users for providing answers. To assist with the mismatch between different expressions in questions and texts, Lin and Pantel [116] proposed an unsupervised method to retrieve inference rules from question answering text. Different from the aforementioned publications that focus on retrieving related questions, answers, answerers, or inference rules for question answering, we focus on the TAR task by asking "yes" or "no" questions to reviewers to locate the missing relevant documents in this chapter.

Similar to our work, Wen et al. [193] proposed a Sequential Bayesian Search algorithm for solving the problem of efficiently asking questions in an interactive search setup. They learn a policy that finds items in a collection using the minimum number of queries. Then Kveton and Berkovsky [107, 108] proposed a generalized linear search (GLS) method to combine generalized search and linear search in recommender systems. The SBSTAR algorithm [224] that we propose differs from the aforementioned work by being applied to unstructured text for the ranking of documents using entities to construct the questions to ask. Further, it updates the model after each question rather than updating by each episode for locating the target item. Moreover, the SBSTAR algorithm is used to find the last few relevant documents in TAR by asking "yes" or "no" questions to reviewers. In this chapter we also extend SBSTAR [224], by incorporating the user's erroneous answers to the SBSTAR model by introducing a new objective function. Further, it provides an analysis on the basis of user simulations. For the user simulations three rudimentary noisy answer models are introduced. Also, we propose an extension of the algorithm, $\text{SBSTAR}_{\text{ext}}$, to decide when to stop asking questions automatically. Last, we conduct a small user study to validate the assumptions made regarding the users' willingness to answer a number of questions, their efforts, and their noisy answers.

Figure 2.2: Pipeline of our approach.



2.3 Methodology

In this section, we first describe in detail the proposed SBSTAR algorithm [224], and in particular the selection of questions asked to a reviewer in a sequential fashion (Section 2.3.2), given the constructed question pool by entity annotation (Section 2.3.1). Then, we present in detail the extensions to the SBSTAR algorithm, and in particular, (a) accounting for reviewers' noisy answers (Section 2.3.3), and (b) introducing an algorithm that decides when to stop asking questions to reviewers (Section 2.3.4).

The pipeline of our approach is shown in Figure 2.2, and a running example is provided in Table 2.1. In particular, while offline, our framework analyzes all the documents in the collection, and generates a pool of CQs to be asked to users. During search time a user submits a query for a certain topic and a search algorithm responds with a ranked list of documents. In our setup, we employ interactive search, using a certain CAL algorithm, the BMI. BMI responds to the user's query with a short ranked list of documents, of a predefined size, and the user is requested to provide relevance feedback over each one of the returned documents. Once feedback is provided the algorithm is trained over it, and produces another ranked list of documents to be shown to the user, and so on. At a certain point, the BMI algorithm decides to stop returning documents, assuming that there is no other relevant document in the collection, or that the effort to find the last few relevant documents is too high. After this stopping point, our algorithm is run to ask CQs. The BMI algorithm is based on a probabilistic classifier (logistic regression), hence at its stopping point a probability over relevance can be calculated for every document in the collection. This probability is used to construct a prior belief of the user's interest over all documents in the collection. Based on this prior belief, our model selects a CQ to ask to the user by picking the entity that best

Table 2.2: Notation.

Notation	Explanation
$\mathcal{D}$, $\mathcal{E}$	document set, entity set
$\mathcal{D}^*$	target document set, $\mathcal{D}^* \subseteq \mathcal{D}$
d, e_l	document $d \in \mathcal{D}$, selected entity in the l-th question $e_l \in \mathcal{E}$
$e_l(\mathcal{D}^*)$	the reviewer reply indicating whether or not the target documents contain the entity e_l
$\pi_l^*(d)$	probability distribution of reviewer preferences over the document d during the l-th question
$\mathbb{P}$	prior belief over the reviewer preferences π^*
α	the Dirichlet parameter of $\mathbb{P}$
N_q	the number of questions to be asked
U_l	the candidate document space during the l-th question
$Z_l(d)$	indicator function for model updating
$h(e)$	error rate of reviewer answers for the entity e
β	the tradeoff parameter for $h(e)$
$D_{training}$, $D_{testing}$	training set, testing set for the when-to-stop classifier
Max_q	the max number of questions to be asked for the when-to-stop classifier

splits the probability mass of the user's interest over the document in the collection into two halves. The algorithm receives the user's answer and on the basis of this answer, the prior belief is updated to a posterior. This posterior belief is used now by our model to select the next query to ask to the user. In the meantime, a classifier is employed to decide whether indeed a next CQ should be asked to the user or the algorithm should produce the final ranked list of the remaining of the documents in the collection.

2.3.1 Question Pool Construction

Entities are recognized as the most vital source of information in text [58, 153]. Based on this assumption we construct the pool of questions using entity annotation, thus focusing on generating questions regarding the presence or absence of an entity in relevant documents. That is, we instantiate the question candidate set by identifying entities in the related documents by using TAGME [61], an entity linking algorithm [47, 117, 118], which has been widely used in prior research [57, 81, 137, 198, 199, 224]. No filters on annotation score (a confidence score for a word or phrase being annotated as an entity) are used for TAGME's results, i.e., all annotations are being considered, which is also a widely used setting in previous work [147, 198, 224]. One can also easily combine TAGME entities with labeled topics [221–223], keyword extraction [23] or other information extraction [126, 149] for the annotations. In this chapter, we take a rudimentary approach by using TAGME to annotate entities in documents, and represent documents by embedding them in the entity space. An example of an annotated

Algorithm 1: SBSTAR [224]

input : A document set, $\mathcal{D}$, the set of annotated entities in the documents, $\mathcal{E}$, a prior belief over document relevance, $\mathbb{P}_0$, and a number of questions to be asked, N_q

foreach *topic* **do**
  $l \leftarrow 1$
  while $l \leq N_q$ **do**
    Compute the reviewer preference:
      $\pi_l^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] \quad \forall d \in \mathcal{D}$
    Use GBS to find the optimal target entity:
      $e_l = \arg\min_e |\sum_{d \in \mathcal{D}} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d)|$
    Ask the question about e_l and observe the reply $e_l(\mathcal{D}^*)$
    Remove e_l from the entity pool
    $l \leftarrow l + 1$
    Update the system's belief $\mathbb{P}_l$ using Bayes' rule:
      $\mathbb{P}_{l+1}(\pi) \propto \pi(d)\mathbb{P}_l(\pi) \quad \forall \pi$
  end
end

document can be seen in Table 2.1. Each one of the entities annotated in this example document, as well as all other documents in the collection, is used to create a question pool. A subset of this pool can also be seen in Table 2.1. After that, the algorithm asks a sequence of questions of the form: "Are the documents about [entity]?" to find the reviewer's target documents. The question template here is ad-hoc and one can also use other defined templates. The reviewers can respond with a "yes", a "no", or a "not sure", with "not sure" ensuring that the reviewers are not forced to give the wrong answer when they are not sure about it.

2.3.2 Sequential Bayesian Search for TAR

The SBSTAR algorithm [224] is described in Algorithm 1. The notation used throughout the paper is summarized in Table 2.2. The input to the algorithm is the document collection, $\mathcal{D}$, the set of annotated entities in the documents, $\mathcal{E}$, a prior belief, $\mathbb{P}_0$, which we model as a Dirichlet distribution parameterized by α, and the number of questions to be asked, N_q. The document set $\mathcal{D}$ is built by running a Boolean query against a biomedical article collection, e.g., PubMed, which constitutes the current approach taken by experts when working on systematic review [93]. This document collection is further reduced by removing the documents that are already discovered by BMI and labeled by experts.

The algorithm assumes that there is a target document set $\mathcal{D}^* \subseteq \mathcal{D}$ the reviewer is interested in, which consists of the last few relevant documents missed by BMI (e.g., the single missing relevant document with ID #20473827 in our running example). We also assume there is a probability distribution modeling the preference of the reviewer over the documents, π^*, over $\mathcal{D}$, and the target documents are drawn i.i.d. from

this distribution. Further, we assume that we have a prior belief $\mathbb{P}$ over the reviewer preferences π^*, which is a probability density function over all the possible realizations of π^*. The system updates its belief when the system observes a reviewer's answer of a question, which is sampled i.i.d. from π^*. At each interaction round l, the reviewer preference $\pi^*_l(d)$ is calculated based on the system's prior belief $\mathbb{P}_l$ over π^*:

$$\pi^*_l(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)], \forall d \in \mathcal{D}. \tag{2.1}$$

Then, the algorithm uses GBS to find the entity, e_l, that best dichotomizes the probability mass of the predicted document relevance, we ask whether the entity e_l is present in the missing target documents that the user wants to find, observe the reply $e_l(\mathcal{D}^*)$, and remove e_l from the entity pool.

In this chapter we consider two settings, regarding the user answers to the system questions. In the first setting, similar to Cormack and Grossman [35], Cormack and Lynam [43], Drucker et al. [51], Roegiest et al. [152], we use the assumption that the human is infallible, and will answer the questions correctly, i.e., he/she will respond with $e(d) = 1$ if the document d contains the entity, while $e(d) = 0$ if the document d does not. The entity in the question is from the entity pool, which is extracted from the corpus. The selection of entities is sequential, that is, we choose an entity to ask a question on while taking into consideration all the previous entities chosen and the corresponding answers. This GBS strategy considers a generalized form of binary search based on the reviewer's preference on document relevance $\pi^*_l(d)$. It chooses the entity that is the most discriminative at each step, which is the one that can split the expected accumulated reviewer's preference $\pi^*_l(d)$ closest to two halves. In particular, it selects the entity with a minimal question selection score by the following objective function:

$$\Big| \sum_{d \in \mathcal{D}} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d) \Big|, \tag{2.2}$$

where $\{e(d) = 1\}$ is either 0 or 1, and thus the term of $(2\mathbb{1}\{e(d) = 1\} - 1)$ is either -1 or 1. In our running example, the first question the system asks is about "[study]", since its question selection score is 0.09 and it is the smallest among all question scores calculated in the first iteration. The reviewer preferences $\pi^*_l(d)$ will be updated by each question and answer, and so will the GBS-based question selection.

After that, the system's belief $\mathbb{P}_l$ is updated using Bayes' rule. The reviewer preference π^* is a multinomial distribution over documents $\mathcal{D}$, hence we model the prior, $\mathbb{P}_0$, by the conjugate prior of the multinomial distribution, i.e., the Dirichlet distribution, with parameter α. In principle, the prior belief $\mathbb{P}_0$ based on α can be set by using any retrieval algorithm. In this chapter, the prior belief $\mathbb{P}_0$ is initialized as $Dir(\alpha)$, with the initial α computed by using the probability of a document being relevant provided by the CAL trained logistic regression; i.e., $\alpha(d) = Pr(d = rel), \forall d \in \mathcal{D}$. Further, we define the indicator vector $Z_l(d) = \mathbb{1}\{e_l(d) = e_l(\mathcal{D}^*)\}$, where $\mathcal{D}^*$ represents the target documents, and $e_l(\mathcal{D}^*)$ is 1 if e_l is present in all the documents in $\mathcal{D}^*$. Intuitively, $Z_l(d)$ is 1 if the entity e_l is both in the target documents and in d, or if it is neither in the target documents nor in d. From Bayes' rule, the posterior belief at the beginning of question

l is:

$$\mathbb{P}_l = Dir(\alpha + \sum_{j=0}^{l-1} Z_j). \tag{2.3}$$

From the properties of the Dirichlet distribution, then we have:

$$\pi_l^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] = \frac{\alpha(d) + \sum_{j=0}^{l-1} Z_j(d)}{\sum_{d' \in \mathcal{D}} (\alpha(d') + \sum_{j=0}^{l-1} Z_j(d'))}, \tag{2.4}$$

where $\alpha(d)$ is the i-th entry of α which corresponds to document d. And thus the reviewer preference π_l^* can be updated by counting and re-normalization. After the last question has been being asked, the system generates the relevance ranked list based on the reviewer preference $\pi_{N_q}^*$ over the documents in the collection that have not been presented by TAR.

2.3.3 Accounting for Noisy Answers

In Algorithm 1, SBSTAR makes the assumption that reviewers, when presented with an entity, know with 100% confidence whether the entity appears in the target documents. To relax this assumption we propose a noise-tolerant version of the algorithm (**E3**). That is, we allow the user to make mistakes and provide the algorithm with wrong answers. In this chapter, we assume that user mistakes are related (in different ways) with the entity the question is asked about, and we model this by $h(e)$, which models the probability that the user will give the wrong answer to a question about entity e. We integrate $h(e)$ into the new objective function, at line 7 of Algorithm 1:

$$e_l = \arg\min_e \left| \sum_{d \in \mathcal{D}} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d) \right| + 2\beta * h(e), \tag{2.5}$$

where β trades off $h(e)$ with the probability mass. The term $h(e)$ is defined in the range from 0 to 0.5, where the highest error rate of 0.5 means that the expert reviewer gives random answers. The first term on the left side of the plus sign, the GBS strategy term, ranges from 0 to 1. To ensure that both terms are in the same range, we multiply $h(e)$ by 2, which enhances clarity in case of manual inspection, even though it is not necessary. The precise definition of $h(e)$ will be provided in Section 2.4. After observing the noisy answer we update the posterior system belief.

2.3.4 When to Stop Asking Questions

In SBSTAR [224] it was observed that a number of questions are effective to identify the last few relevant documents. However, different queries, and different stopping points of the CAL algorithm may require a different number of questions to be asked. In this section, we describe our question stopping method. Different from defining the number of questions in advance, we propose an extension of SBSTAR that we call SBSTAR$_{ext}$, which explores an automatic method of determining when to stop. In particular, SBSTAR$_{ext}$ trains a classifier based on a number of extracted features to

Table 2.3: The defined features for deciding when to stop asking questions.

Feature	Description
Stopping point	The percentage of documents reviewed through CAL
# of yes/no questions	The number of questions asked which got yes or no feedback from reviewer
yes/no answer	Whether the last question asked got yes/no answer from reviewer or not
# of candidates	The number of documents in the current user space, split by GBS
difference in # of candidates	The reduction of the number of documents in the current user space compare with last user space
difference in preference	The change of user preference π^* in term of last user preference π^*
difference of top 1	Whether the highest ranked document changes or not

dynamically decide whether to stop or continue to ask questions, at every interactive round. After examining which factors could affect the decision of when to stop asking questions, we define seven dynamic features including the CAL "Stopping point", the "# of yes/no questions" asked so far, whether the last question received an answer that "yes/no answer", the "# of candidate(s)" documents left, the "difference in # of candidates", the "difference in (user) preference", and the "difference of top 1" ranked document.[3] The features we defined are shown in Table 2.3. The intuition behind using these features is the following: different stopping points for BMI may affect the number of missing documents and thus affect the number of questions to locate the last few relevant documents. The features "# of yes/no questions" and "yes/no answer" also affect when to stop asking questions since having asked many questions already may indicate that it is time to stop, while receiving a "not sure" answer as opposed to "yes" or "no" indicates that no further information was obtained in this last round. As for "# of candidate(s)" documents left and "difference in # of candidates", a smaller number of documents left or smaller difference in the number of candidates before and after asking a question indicates a reduction in the space of possible documents and hence hints at higher chances to stop asking questions. The features "difference in (user) preference" and the "difference of top 1" measure the change in user preference and in the top document respectively, and thus provide a signal of whether eliciting further user preferences may make a difference or not in the ranking and thus whether it is time to stop asking questions. This is by no means an exclusive set of useful features and more could be engineered, however our experiments indicate that these features are effective enough. In this chapter, we use the SVM, random forests, and feedforward neural network based classifiers, and show a performance comparison in Section 2.4.5.

[3] An external experiment is conducted. We first define as many factors as possible which may affect when to stop asking questions, and then we filter out these factors which have very small or no influence for when to stop asking questions by performance experiments, at the end seven features are left.

Algorithm 2: SBSTAR$_{\text{ext}}$

input : A document set, $\mathcal{D}$, the set of annotated entities in the documents, $\mathcal{E}$, a prior belief over document relevance, $\mathbb{P}_0$, and the max number of questions to be asked, Max_q

Training a classifier using training set:
  $Classifier(D_{training})$
foreach *topic* $\in D_{testing}$ **do**
  $l \leftarrow 1$
  $Stop \leftarrow$ False
  while $l \leq Max_q$ ***and*** $Stop = False$ **do**
    Compute the reviewer preference:
      $\pi_l^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] \quad \forall d \in \mathcal{D}$
    Use GBS to find the optimal target entity:
      $e_l = \arg\min_e |\sum_{d \in U_l}(2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d)|$
    Ask the question about e_l and observe the reply $e_l(\mathcal{D}^*)$
    Remove e_l from the entity pool
    Reduce the set of documents of candidate version space U_l:
      $U_{l+1} = U_l \cap \{i \in \mathcal{D} : e_l(i) = e_l(\mathcal{D}^*)\}$
    $l \leftarrow l + 1$
    Update the system's belief $\mathbb{P}_l$ using Bayes' rule:
      $\mathbb{P}_{l+1}(\pi) \propto \pi(d)\mathbb{P}_l(\pi) \quad \forall \pi$
    Compute the features of Table 2.3
    Predict the label of the trained classifier $Pred_{classifier}$ using the computed features:
    $Stop = Pred_{classifier}$
  end
end

The SBSTAR$_{\text{ext}}$ algorithm is presented in Algorithm 2. Let $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ be a training data set, in which x_i denotes a question instance (each question is represented by a seven-feature vector) and $y_i \in \{\text{“}True\text{”}, \text{“}False\text{”}\}$ denotes a classification label. For each query (i.e., topic) in the training space $D_{training}$, which contains all of the training documents for this query, we sequentially ask 100 questions (on the basis of the SBSTAR algorithm), and label each question as “True” or “False”. “True” means stop asking, while “False” means continue asking. To decide whether to label a question as “True” or “False”, we look at the ranking of documents produced by the SBSTAR algorithm after each question has been asked. There is a point (question) after which the position of the target relevant documents does not change anymore. All questions up to that point are labeled as “False”, while all the following up to 100 questions are marked as “True”. That means we have 100 question instances and assign 100 labels for each topic-stopping point pair.

We first train a when-to-stop classifier using the training set. During testing time, for each document in $D_{testing}$, we first compute the user preference using the belief $\mathbb{P}_l$, and

find the optimal entity e_l that best splits the probability mass of the predicted document relevance. We ask whether the entity e_l is present in the missing target documents that the user wants to find, observe the reply $e_l(\mathcal{D}^*)$, and remove e_l from the entity pool. Then we reduce the candidate document space, $U_l \subseteq \mathcal{D}$, and update the system's belief $\mathbb{P}_{l+1}$ using Bayes' rule. After that, we compute the features of Table 2.3 and predict the output of the when-to-stop classifier by using the pre-trained classifier model. If the output of the trained classifier model using the dynamically computed features is "False", the SBSTAR$_{\text{ext}}$ system continues by selecting the next question to ask and updates the posterior. If the output of the trained classifier model is "True", we stop asking questions and generate the final recommended relevant documents ranked list.

2.4 Experiments and Analysis

In this chapter we decompose **RQ1** into the four following subquestions and aim to answer them:

RQ1.1 How does the stopping point of CAL, as well as the number of questions asked by SBSTAR affect the performance of the algorithm?

SBSTAR is affected by two parameters: (a) the stopping point of the CAL algorithm; if CAL stops too early it may be harder to locate all the remaining documents; and (b) the number of questions asked by SBSTAR. We will explore the effect on the model performance of varying stopping points and the number of asked questions.

RQ1.2 How effective is SBSTAR in finding the remaining relevant documents compared to the baselines?

We compare the SBSTAR model with three different baselines (see Section 2.4.1 in detail). This research question is used to confirm the effectiveness of the SBSTAR model, and investigate the extent to which the SBSTAR model outperforms state-of-the-art methods.

RQ1.3 What is the influence of noisy answers on the SBSTAR performance?

We investigate the effect of reviewers' noisy answers on the performance of our algorithm. We consider three noise settings: one with a fixed error rate for all of the questions (entities), one for which the error rate is a function of the term frequency of the entity, and one is defined on the basis of target documents. This research question explores the robustness of our SBSTAR model to noise.

RQ1.4 Is our proposed method for deciding when to stop effective?

Previous work needed to define the number of asked questions as an input parameter to the algorithm. We investigate how dynamically deciding when to stop asking questions performs.

2.4.1 Experimental Setup

Dataset. The dataset used in the experiments is the collection released by the Technology Assisted Reviews in Empirical Medicine Task of the CLEF 2017 e-Health Lab[4] [66, 93], which is also well adopted by other work [41, 50, 110, 162]. The collection contains 50 topics, and $266,967$ abstracts of MEDLINE articles identified by

[4]https://sites.google.com/site/clefehealth2017/task-2

Table 2.4: The number of missing documents in different stopping points on the abstract level (top) and the document level (bottom).

Stopping point	0%	10%	15%	20%	25%	30%	35%	40%
# of missing docs	4661	1225	758	485	311	203	134	81
Stopping point	45%	50%	55%	60%	65%	70%	75%	80%
# of missing docs	52	35	24	12	4	3	3	3

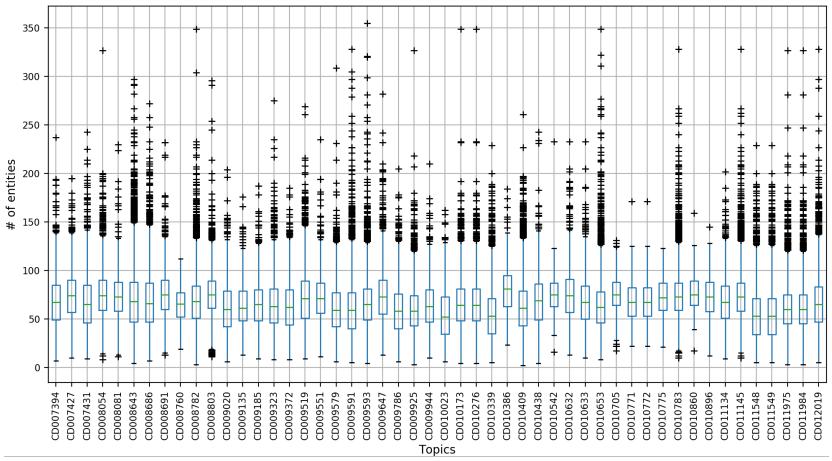
Stopping point	0%	10%	15%	20%	25%	30%
# of missing docs	1093	123	56	23	10	5
Stopping point	35%	40%	45%	50%	55%	60%
# of missing docs	5	3	3	2	2	0

PMID, and the relevance judgments for each of these articles against the 50 topics, both at an abstract and at a document level. Each topic file is in a text format and contains four sections: topic ID, topic title, the query, and a list of PMIDs of documents. The query corresponds to the Boolean query used to obtain the PMIDs relevant to the given topic, which need to be re-ranked. Each document file linked by PMID is in the XML format and contains the titles, abstracts, and metadata for an article. Firstly, we use the java SAX parser to extract the title and abstract of XML files. After removing documents without abstract we ended up with $221,654$ documents. The remaining preprocessing steps include tokenization, elimination of stop-words, stemming and case unification. For each topic the relevant documents were also provided. In systematic reviews there are typically two levels of relevance judgments. The first is at the abstract level: the expert submits a Boolean query and examines the titles and abstracts of the returned set, judging whether these returned abstracts summarize potentially relevant articles. That is, the expert provides a relevance label for each article ID returned by the query based on the abstract. The second is at the full text level, where the full document that corresponds to the previously identified relevant abstracts is read and the relevant ones are identified. That is, the expert refines the relevance label for those article IDs that were assigned a positive label before at the abstract level.

For the first three research questions **RQ1.1**–**RQ1.3**, we test our model over all of the 50 topics. For **RQ1.4**, same with the Technological Assisted Reviews in Empirical Medicine Task[5] [93] in CLEF 2017[6] and other works [41, 50, 110, 162], we use 20 topics as the training set and test our model on the remaining 30 topics. The number of missing documents in different stopping points on the abstract level (top) and the document level (bottom) is shown in Table 2.4. The five-number summary of entities for each topic is shown in Figure 2.3. From Figure 2.3, we can see that different topics have a similar five-number summary trend. The medians of the number of entities in different topics are between 50–80.

[5]https://sites.google.com/site/clefehealth2017/task-2
[6]http://clef2017.clef-initiative.eu/index.php

Figure 2.3: The five-number summary of entities for each topic.



Evaluation measures. We use two evaluation metrics that were the official metrics in CLEF 2017 e-Health Evaluation Lab [93]: Mean Average Precision (MAP) and last_rel, which is the position of the last relevant document in the ranking, which approximates the user effort made, in terms of documents that need to be reviewed, to find all relevant documents in the collection:

$$MAP = \frac{\sum_{q=1}^{Q} AP(q)}{Q}, \tag{2.6}$$

where Q is the number of topics in the testing dataset, in our experiments, Q is 50. $AP(q)$ is the average precision of the topic q:

$$AP = \frac{\sum_{l=1}^{n} P(l) * rel(l)}{\text{\# of relevant documents}}, \tag{2.7}$$

where n is is the number of documents in the ranked list, l is the rank in the sequence of retrieved documents, $P(l)$ is the precision until l, i.e., the number of relevant documents out of the top-ranked documents. $rel(l)$ is the ground truth capturing whether the document is relevant to this topic.

Simulating reviewers. The experiments depend on the ability of the reviewers to answer the questions asked to them by our model. In this chapter we simulate users following past work in interactive algorithms [213, 219]. We also conduct a small user study described in Section 2.4.6. We simulate users under two different settings:

(1) we assume that the user will respond to the questions knowing precisely whether an entity appears or not in the missing relevant documents. Here the user model assumption is that the reviewer has an initial target document set in mind, which is deterministic but unknown. If an entity is contained in all missing relevant documents, then the reviewer will respond with a "yes" answer, if an entity is

absent with a "no" answer, and for anything in between, with a "not sure" answer. This setting is the same with Zhang et al. [213], who assume that the user fully knows the value of the question on an aspect;

(2) we allow the users to give the wrong answer to our system with a given probability.

In the latter case, we consider three noisy answers settings, regarding the error rate for each entity $h(e)$:

(a) In the first setting all entities have the same chance to invoke a wrong answer and hence $h(e)$ is equal across entities; in this case, we experiment with different error rates that range from 0.1 to 0.5 with a step of 0.1. An error rate $h(e)$ of 0.5 means that the user has a 50% probability to give the wrong answer (i.e., randomly give the answer).

(b) In the second setting we assume that users are more confident in their answers about an entity e if e is frequently occurring in the documents related to a given topic (query), and we define $h(e)$ as a function of average term frequency (TF) of an entity e across all documents, which lies in the range of (0, 0.5]:

$$h(e) = \frac{1}{2(1 + TF_{avg}(e))}. \tag{2.8}$$

$TF_{avg}(e)$ represents the average term frequency of an entity e across all documents related to a given topic. In our experiments this subset of documents related to a topic is provided to us by the way the collection has been constructed, but one could think of other heuristics to define topic-related documents (such as running a ranking function, e.g., BM25, and considering the top-1000 documents).

(c) In the third setting the noisy answers are modeled by multi-target property. We assume that the reviewer is more confident about the entity that tends to appear concurrently in all of the missing relevant documents. In this case $h(e)$ is also in the range (0, 0.5] and is defined as:

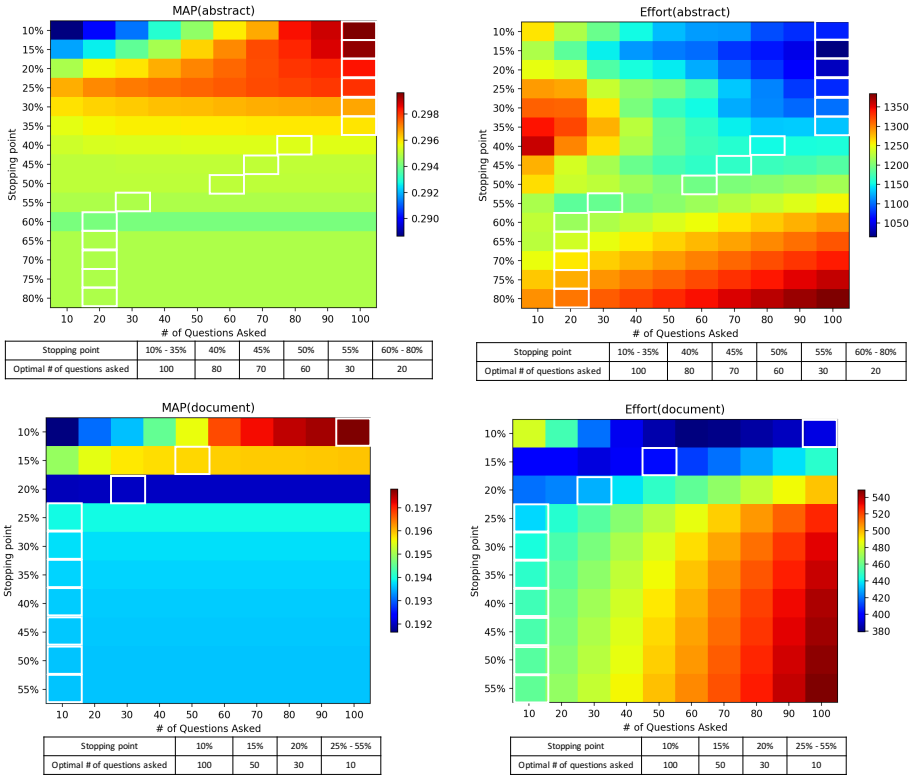
$$h(e) = \frac{\min(N_{\{e(\mathcal{D}^*)=1\}}, N_{\{e(\mathcal{D}^*)=0\}})}{N_{\{e(\mathcal{D}^*)=1\}} + N_{\{e(\mathcal{D}^*)=0\}}}. \tag{2.9}$$

$N_{\{e(\mathcal{D}^*)=1\}}$ represents the number of target documents containing entity e for a given topic, while $N_{\{e(\mathcal{D}^*)=0\}}$ represents the number of target documents that do not contain entity e for a given topic.

In all three settings, and during simulations, once it is decided that a wrong answer will be provided to the system, the simulator will randomly choose a wrong answer out of two wrong answers available.

Note that the selection of these three noise settings is ad-hoc; the error can be defined as any other function of any other characteristic of entities, reviewers or topics. One should conduct a large user study to identify how and why users give erroneous answers to such system questions, but we leave this as future work.

Figure 2.4: Heatmap of the MAP and the total effort required to achieve total recall on the abstract level (top) and the document level (bottom). The total effort is naively defined as the sum of the rank of the last relevant document and the number of asked questions. For MAP, the more red the heat map, the better the model performance. For effort, the more blue the heat map, the better the performance. The optimal number of questions for the corresponding stopping point is designated with the white boundary box and the tables below.



Stopping point	10% - 35%	40%	45%	50%	55%	60% - 80%
Optimal # of questions asked	100	80	70	60	30	20

Stopping point	10% - 35%	40%	45%	50%	55%	60% - 80%
Optimal # of questions asked	100	80	70	60	30	20

Stopping point	10%	15%	20%	25% - 55%
Optimal # of questions asked	100	50	30	10

Stopping point	10%	15%	20%	25% - 55%
Optimal # of questions asked	100	50	30	10

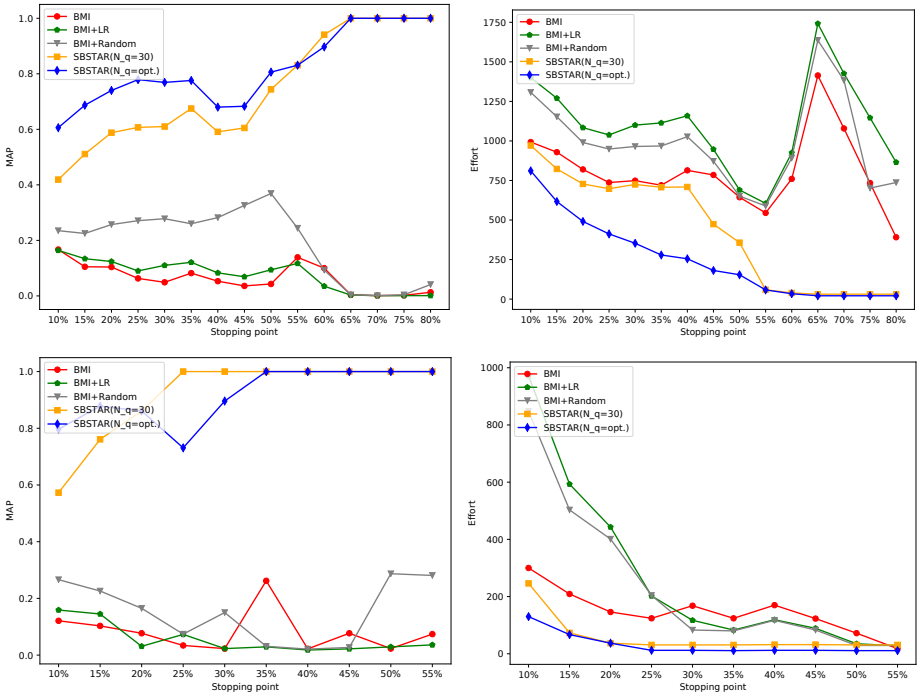
Baselines. We compare our method to three baselines, (1) **BMI** [41], which is the state-of-the-art CAL algorithm applied without any stopping criterion until the entire collection is reviewed, (2) **BMI + LR**, which applies BMI until a number of documents are reviewed (stopping point) and then ranks the remaining of the collection on the basis of the trained logistic regression model, and (3) **BMI + Random**, which applies BMI until a number of documents are reviewed (stopping point) and then randomly chooses entities to ask about.

2.4.2 **RQ1.1** The Effect of the CAL Stopping Point and the Number of Questions

The SBSTAR algorithm is parameterized by (a) the point at which BMI stops providing documents to reviewers for relevance feedback, and (b) the number of questions about entities asked consequently to the reviewer. To better understand their impact on the performance of the model, we report the MAP and total effort required to reach 100% recall on the abstract level and the document level. Figure 2.4 shows the heatmap of the MAP (left) and the total effort (right) required to reach 100% recall on the abstract level (top) and the document level (bottom). The x-axis is the number of asked questions ranging from 10 to 100, and the y-axis is the stopping point as a percentage of the collection shown to the reviewer by BMI. We measure the effort on the basis of two indicators: (a) the total number of documents required to be reviewed to reach 100% recall (i.e., the last_rel measure); this consists of both the documents ranked by BMI before the stopping point and the documents ranked by SBSTAR after the stopping point; and (b) the number of questions asked by SBSTAR. The effort is calculated as the sum of the two numbers, by making the simplifying assumption that answering a question takes the same effort as judging the relevance of a document. From this figure, we can see the visualization of the evolution of the MAP and effort over the number of asked questions, and we can also see the trends of MAP and effort across different stop rates.

We observe that the MAP is always increasing or stable with the number of questions increases, although at different stop rates, for both the abstract level and the document level. It is obvious that we get more accurate results when we ask the reviewer more questions. And the overall trends of MAP over different stop rates are increasing and then drop down in the early stage (less than 80 questions for the abstract level and less than 60 questions for the document level respectively). It means that the stop rate should be set as a suitable value, should not be too high or too low. The MAP always displays a decreasing trend in the late stage (greater than or equal to 80 questions for the abstract level and greater than or equal to 60 questions for the document level, respectively). This might be because that SBSTAR can always get a good relevance ranked list when the number of asked questions is high and the ranked list before the stop rate (i.e., is equal to the ranked list of BMI) lowers the mean of average precision. The highest MAP is achieved when the stop rate is 10% and the number of questions is 100. We also observe that the overall trend of effort grows with the number of questions asked when the stopping point is greater than 55%, while the effort is decreasing when the stopping point is less than 55% for the abstract level. As for the document level, the overall trend of effort increases with the number of questions asked when the stopping point is greater than 10%, while the effort decreases when the stopping point is equal to 10%. This might be because the missing relevant documents are very few when the stopping point has a high value, in which case asking many questions only leads to higher effort. Furthermore, SBSTAR(N_q=opt.) can reduce the effort effectively when the stopping point is less than or equal to 50% for the abstract level, while the stopping point is equal to 10% for the document level, respectively. The effort fluctuates over different stopping points and the effort is relatively lower when the stopping point is between 15% and 20%, and between 45% and 55% for the abstract level, while it is relatively lower when

Figure 2.5: Performance comparison measured by MAP and total effort with different stopping points (the percentage of documents to be reviewed through BMI) on the abstract level (top) and the document level (bottom). N_q=opt. stands for N_q=optimal. The near-optimal number of questions were asked by SBSTAR ($N_q = opt.$) and BMI+Random for each stopping point is indicated in Figure 2.4 of **RQ1.1**. SBSTAR performs better than baselines.



the stopping point is between 10% and 15% for the document level, respectively. The lowest effort is achieved when the stopping point is 15% and the number of asked questions is 100 for the abstract level while the stopping point is between 10% and the number of asked questions is 60 for the document level, respectively.

2.4.3 **RQ1.2** The Performance of the SBSTAR Method

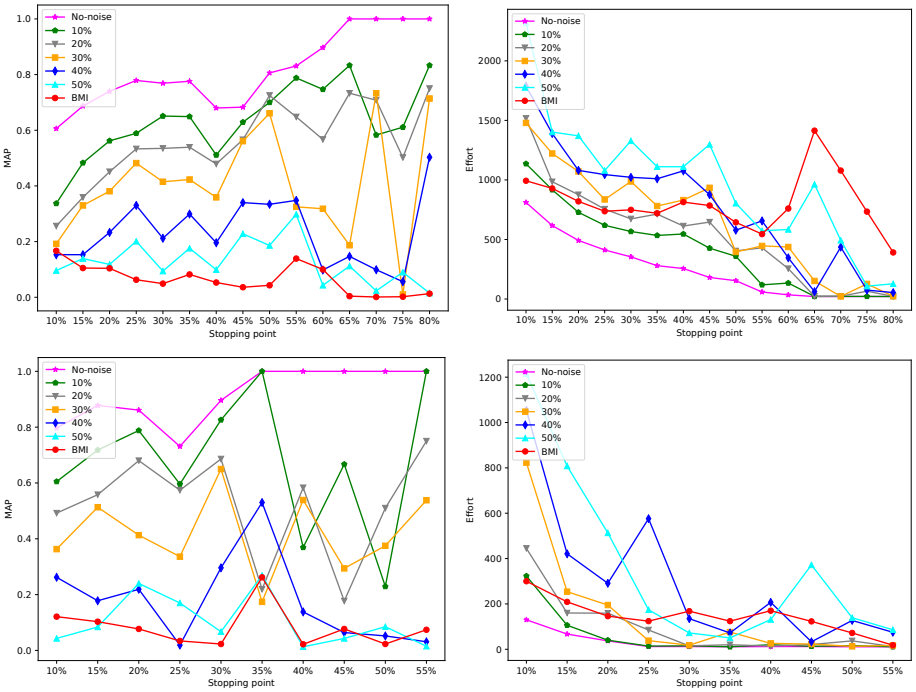
We compare the performance of SBSTAR with the state-of-the-art baselines in this section to explore the effectiveness of the SBSTAR model. In this research question, different from **RQ1.1**, MAP and total effort are computed only on the basis of the documents ranked after the stopping point, since we want to isolate the model effectiveness. The three compared baselines are shown in Section 2.4.1. Figure 2.5 shows the comparison results measured by MAP and effort on the abstract level and the document level. The values with the best performance are shown in boldface. As indicated in Figure 2.5, the SBSTAR algorithm achieves the highest results when compared to the

three baselines. The SBSTAR algorithm outperforms BMI, BMI + LR, and BMI + Random on both MAP and effort. When considering the abstract level relevance labels, the results show that the SBSTAR model can improve BMI by 0.439 to 0.999 points on MAP, it can improve BMI + LR by 0.442 to 0.999, and it can improve BMI + Random by 0.357 to 0.999. Note again that MAP is measured against the missing documents, that we consider the ranking to start after the BMI stopping point. The results show the SBSTAR model can relatively reduce the effort by 18.3% to 98.5%, 42.1% to 98.8% and 38% to 98.7% compared with BMI, BMI + LR, and BMI + Random respectively. When considering the document level labels, the results show that the SBSTAR model can improve BMI by 0.676 to 0.978 MAP points, it can improve BMI + LR by 0.638 to 0.982, and it can improve BMI + Random by 0.531 to 0.979. Again, MAP is measured against the missing documents; we consider the ranking to start after the BMI stopping point. The results show that the SBSTAR model can reduce the effort by 38.9% to 92.9%, 60.7% to 94.1% and 62.1% to 94.1%, compared with BMI, BMI + LR, and BMI + Random, respectively. The SBSTAR algorithm outperforms the BMI + Random baseline, which indicates, as expected, that our question selection strategy is better than choosing questions randomly. The SBSTAR algorithm also greatly improves over the BMI and BMI + LR baselines, and even the performance of BMI + Random is superior to that of BMI and BMI + LR. This clearly suggests that a theoretically optimal sequence of entity-centered questions can be rather effective. It is expected that BMI will outperform BMI + LR because of repeatedly training the LR classifier [41]. There are some results that BMI + LR outperforms BMI, especially when the stopping point is less than and equal to 50%. This might be because repeatedly enriching the training data and training LR classifier does not push all of the ranking positions of the missing relevant documents up, but instead by pushing part of the missing relevant documents up and part of them down, since there are multiple missing relevant documents. When the stopping point is larger than 50%, BMI obviously outperforms BMI + LR since there are very few missing relevant documents. Additionally, we add the results for SBSTAR when the number of asked questions is equal to 30, i.e., SBSTAR(N_q=30), to show the performance of the SBSTAR model when asking a fixed smaller number of questions instead of the optimal number of questions. The results show that SBSTAR can still perform better than the state-of-the-art baseline BMI when asking 30 questions. Table 2.9 provides a few examples of a sequence of questions session asked by SBSTAR.

2.4.4 **RQ1.3** The Effect of Noisy Answers

Given that the user may not always know the right answer to a system's question, we also explore the noise-tolerance of the SBSTAR algorithm towards answering **RQ1.3**. We develop a noise-tolerant version of the SBSTAR algorithm as shown in Section 2.3.3 and investigate what the influence of noisy answers is. As explained in Section 2.4.1, we simulate the noisy answer of the user under three settings. In the first setting, we fix the probability of error, and consider it as a parameter that ranges from 10% to 50% with a step 10%. The results when $h(e)$ is a fixed number are shown in Figure 2.6. It is obvious and expected that the overall MAP decreases the probability of error increases while the effort (in terms of the number of documents that need to be reviewed for the reviewer to reach the last relevant document) increases with an increase in the probability of

Figure 2.6: The results with noisy answers in terms of MAP and the total effort required to achieve total recall on the abstract level (top) and the document level (bottom) when $h(e)$ is a fixed, ranging from 10% to 50% with a step 10%. The near-optimal number of questions were asked for each stopping point is indicated in Figure 2.4 of **RQ1.1**. SBSTAR still achieves relatively good performance for a 10% wrong answer rate.



error. When the probability of error is equal to 50%, which means the user answer the question with "yes", "no" or "not sure" at random, we observe very low performance. On the other hand, when the probability of error is equal to 10%, the values of the two metrics still achieve relatively good performance. Further, note that, in comparison to the state-of-the-art baseline, the BMI model, for a 10% wrong answer rate, the MAP of SBSTAR is still higher.

In the second setting, we define $h(e)$ as a function of the average term frequency of the entity e as shown in Equation 2.8. In this case we perform the experiment changing β in the range of 0 to 1 with step size 0.1. The results in terms of MAP when $h(e)$ is modeled by term frequency are shown in Table 2.5 and the results in terms of effort are shown in Table 2.6. Optimal results at different stopping points are shown in bold. As we can see from the table, the performance as measured by the two metrics when using the optimal β is higher than or equal to that when $\beta = 0$, which suggests that the objective function of our noise-tolerant version of SBSTAR algorithm is effective.

In the third setting $h(e)$ is defined on the basis of the target documents as shown in Equation 2.9. The results of this experiment are shown in Figure 2.7. From Figure 2.7,

Table 2.5: The results of noisy answers in terms of MAP on the abstract level (top) and the document level (bottom) when $h(e)$ is modeled by term frequency. β ranges from 0 to 1 with step size 0.1. Optimal results in different stopping points are shown in bold. The near-optimal number of questions for each stopping point is indicated in Figure 2.4. Results with optimal β are better than or equal to that when β = 0, which suggests that the objective function of our noise-tolerant version of SBSTAR algorithm is effective.

Stopping point	No-noise	β=0	β=0.1	β=0.2	β=0.3	β=0.4	β=0.5	β=0.6	β=0.7	β=0.8	β=0.9	β=1
10%	0.606	0.147	0.147	0.175	0.143	0.160	0.124	0.108	0.149	0.105	**0.189**	0.168
15%	0.687	0.179	0.201	0.246	0.235	0.194	0.219	**0.254**	0.224	0.162	0.213	0.217
20%	0.740	0.251	0.257	0.300	0.247	0.222	0.292	0.301	0.241	0.261	0.243	**0.355**
25%	0.779	0.342	0.321	0.325	0.316	**0.403**	0.308	0.391	0.265	0.326	0.329	0.313
30%	0.769	0.289	0.295	0.349	0.272	0.262	0.327	0.328	0.340	0.250	0.345	**0.358**
35%	0.776	0.321	0.300	0.360	0.299	0.317	0.362	**0.401**	0.328	0.326	0.293	0.367
40%	0.680	0.219	0.289	0.359	0.281	0.301	**0.381**	0.298	0.347	0.305	0.282	0.338
45%	0.683	0.446	0.320	0.384	0.362	0.308	0.464	**0.538**	0.437	0.366	0.434	0.507
50%	0.806	0.331	0.379	0.418	0.438	0.465	0.376	**0.566**	0.428	0.478	0.506	0.522
55%	0.831	0.415	0.503	0.500	0.418	0.317	0.436	0.530	0.436	0.299	0.407	**0.532**
60%	0.897	0.180	0.335	0.266	0.580	**0.639**	0.323	0.106	0.375	0.290	0.291	0.243
65%	1.000	0.252	0.026	0.068	0.393	**0.614**	0.342	0.611	0.018	0.117	0.193	0.042
70%	1.000	0.019	0.098	0.513	0.418	**0.667**	0.013	0.181	0.061	0.338	0.142	0.339
75%	1.000	0.041	0.337	0.034	0.212	0.215	**0.375**	0.340	0.371	0.201	0.338	0.169
80%	1.000	0.051	0.195	0.159	0.243	0.118	0.278	0.095	0.335	0.362	**0.459**	0.155

Stopping point	No-noise	β=0	β=0.1	β=0.2	β=0.3	β=0.4	β=0.5	β=0.6	β=0.7	β=0.8	β=0.9	β=1
10%	0.797	0.150	0.274	0.280	0.262	0.275	0.262	0.266	0.237	**0.335**	0.313	0.332
15%	0.878	0.338	0.376	0.311	0.360	0.420	0.336	**0.425**	0.422	0.385	0.416	0.344
20%	0.861	0.419	0.425	**0.560**	0.373	0.379	0.426	0.503	0.399	0.452	0.512	0.367
25%	0.731	0.236	0.223	0.220	**0.475**	0.124	0.375	0.267	0.256	0.336	0.186	0.223
30%	0.896	0.374	**0.637**	0.119	0.275	0.436	0.361	0.181	0.540	0.264	0.317	0.355
35%	1.000	0.515	0.160	0.406	**0.688**	0.513	0.458	0.271	0.183	0.198	0.386	0.360
40%	1.000	0.273	0.066	0.169	0.341	0.569	0.261	0.140	0.569	**0.667**	0.071	0.187
45%	1.000	0.080	0.500	0.113	0.268	0.533	0.029	0.528	0.438	0.369	0.293	**0.613**
50%	1.000	**1.000**	0.505	0.556	1.000	0.507	0.417	0.545	0.750	0.625	0.417	0.545
55%	1.000	**1.000**	0.031	0.512	0.750	0.750	0.313	1.000	0.750	0.200	0.750	0.258

despite the presence of noise, SBSTAR still achieves comparable effort and greatly outperforms the baselines in terms of MAP.

2.4.5 **RQ1.4** The Effectiveness of Stopping to Ask Question

To answer **RQ1.4** we explore the effectiveness of our stopping algorithm $\text{SBSTAR}_{\text{ext}}$. We first perform the experiment using SVM classifier with different maximum numbers of questions Max_q to be asked, from 100 to 500 with step size 100. The results of the stopping algorithm on the abstract level (top) and the document level (bottom) are shown in Figure 2.8. As we can see from Figure 2.8, different values of Max_q generate

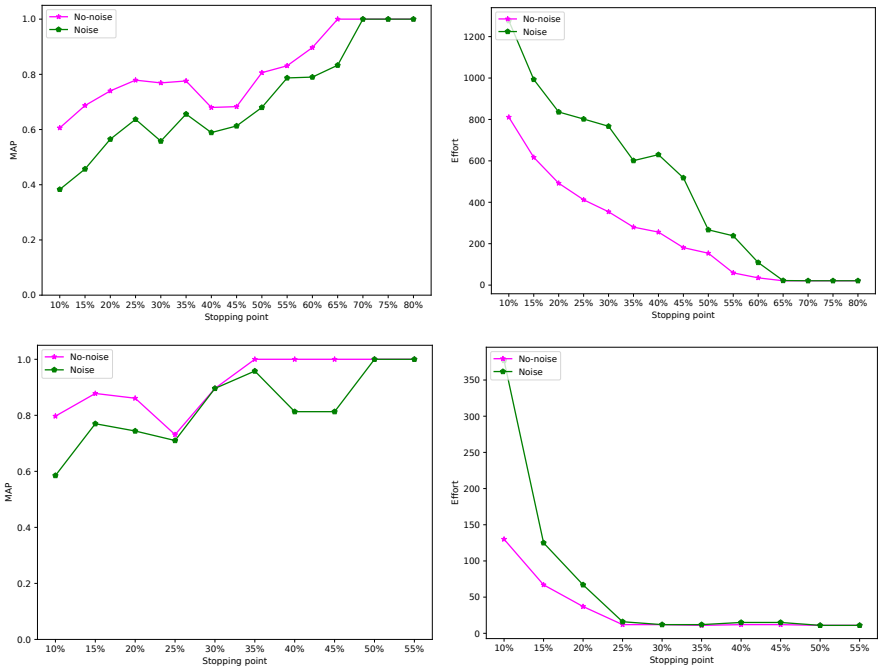
Table 2.6: The results of noisy answers for the total effort required to reach 100% recall on the abstract level (top) and the document level (bottom) when $h(e)$ is modeled by term frequency. β is ranging from 0 to 1 with a step 0.1. Optimal results at different stopping points are shown in bold. The near-optimal number of questions for each stopping point is indicated in Figure 2.4. Results with optimal β are better than or equal to that when β = 0, which suggests that the objective function of our noise-tolerant version of SBSTAR algorithm is effective.

Stopping point	No-noise	β=0	β=0.1	β=0.2	β=0.3	β=0.4	β=0.5	β=0.6	β=0.7	β=0.8	β=0.9	β=1
10%	811	1854	**1665**	1724	1856	1840	1874	2048	1761	1983	1813	1729
15%	617	1585	1423	1484	1349	**1234**	1454	1299	1384	1483	1290	1552
20%	492	1374	**1040**	1155	1090	1119	1124	1194	1122	1080	1248	1124
25%	412	1006	998	1028	**946**	990	1008	1001	1047	1116	1042	1134
30%	354	1053	974	1128	986	**937**	1173	1114	1267	1087	1058	1074
35%	280	**844**	908	974	959	981	971	1056	985	889	916	975
40%	256	1203	1233	1263	1134	1023	1102	1207	1014	**977**	1122	1067
45%	181	1172	1039	966	**814**	863	966	1136	1132	1026	1155	846
50%	154	942	530	704	837	**436**	745	581	651	602	503	869
55%	59	576	427	303	452	354	239	378	300	292	439	**157**
60%	35	365	278	496	**162**	297	529	183	275	366	607	316
65%	21	690	276	105	45	58	76	**23**	120	410	89	94
70%	21	212	54	**30**	134	159	118	302	58	508	31	402
75%	21	51	140	204	57	159	**45**	85	51	51	522	240
80%	21	121	69	34	34	34	212	32	239	144	**29**	32

Stopping point	No-noise	β=0	β=0.1	β=0.2	β=0.3	β=0.4	β=0.5	β=0.6	β=0.7	β=0.8	β=0.9	β=1
10%	130	1078	1056	1013	969	943	1175	1069	986	**889**	963	997
15%	67	325	301	267	284	709	430	478	**267**	402	654	532
20%	37	143	170	**118**	397	168	290	128	286	178	205	252
25%	12	113	42	321	**18**	90	31	35	42	62	69	31
30%	12	35	24	34	41	**19**	50	36	38	32	35	21
35%	11	41	22	16	**15**	83	21	17	20	22	18	24
40%	12	29	44	128	26	17	53	38	18	**13**	36	38
45%	12	103	**13**	23	17	19	150	27	23	18	255	15
50%	11	**11**	66	15	**11**	49	13	16	12	13	13	16
55%	11	**11**	49	31	12	12	15	**11**	12	15	12	43

different results. Obviously, the MAP is increasing with an increase in Max_q for the abstract level, this might be because the increase of Max_q will increase the number of questions asked in some topics. As for the document level, the MAP is increasing at early stages, and eventually stay stable with the increase of Max_q. This might be because the increase of Max_q will increase the number of questions asked in some topics in the beginning, and finally stops increasing since the Max_q is set to too high so that the number of questions in all of topics is less than the value of Max_q. The total effort required to achieve 100% recall fluctuates with different values of Max_q when the stopping point is low and then stays stable when the stopping point is set

Figure 2.7: The results of noisy answers in terms of MAP and the total effort required to reach total recall on the abstract level (top) and the document level (bottom) when $h(e)$ is defined by using multi-target property. The near-optimal number of questions for each stopping point is indicated in Figure 2.4.



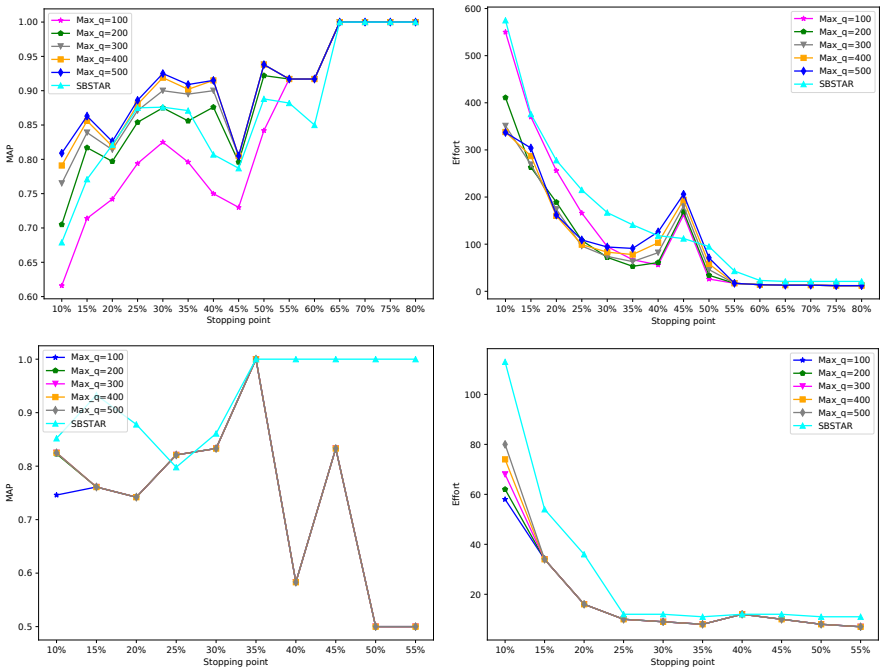
to a high value. When $Max_q = 300$ for the abstract level, most of MAP and effort are superior than original SBSTAR. The average MAP at different stopping points is improved by 3.5% and the average effort is improved by 36.4%. Note that, the original SBSTAR uses the optimal number of asked questions by grid search in **RQ1.1**. As for the document level, we can see that $\text{SBSTAR}_{\text{ext}}$ for deciding when to stop asking questions automatically still achieves a very high MAP while greatly reduces the effort. When $Max_q = 300$, the average MAP at different stopping points is reduced by 20.6% while the average effort is improved by 36.1%.

We also compare the performance of different classification models, including SVM, random forests (RF), and feedforward neural network (NN). Here, we use the optimal value of Max_q from the previous experiment, 300, in Figure 2.8. The MAP and total effort at the abstract level (top) and the document level (bottom) are shown in Table 2.7. The average number of asked questions is also shown in parentheses following the total effort. For the comparison, the WEKA API and its default model parameters[7] are used.[8]

[7] https://www.cs.waikato.ac.nz/ml/weka/index.html

[8] We use the WEKA API and default model parameters in WEKA to perform the classifier models, i.e., for SVM, C = 1.0, and kernel = RBF; for random forests, the max number of trees = 100; for neural network, learning rate = 0.3, momentum = 0.2, and number of hidden neurons = 4.

Figure 2.8: The results of when to stop on the abstract level (top) and the document level (bottom) with different maximum number of asked questions. The near-optimal number of questions for each stopping point for SBSTAR is indicated in Figure 2.4.



As we observe in Table 2.7, mostly the SVM and the random forest achieve higher MAP and a lower effort than the neural network. That is, the SVM and the random forest classifiers perform better. When the stopping point is set to a high value (here: the stopping point is greater than or equal to 65% on the abstract level, and greater than or equal to 40% on the document level), there are some peaks and the performance in terms of MAP fluctuates strongly. This might be because all of the relevant documents are successfully found for most of topics and there are very few relevant documents remaining (here: less than 5 relevant documents in total). We conduct a further analysis to understand the relative importance of different features in predicting when to stop. Similar to Genuer et al. [65], we use the trained random forests to inspect the relative feature importance. The results of feature importance computed by random forests on the abstract level(top) and the document level(bottom) are shown in Table 2.8. From Table 2.8, we can see the relative feature importance on the abstract level and the document level respectively. The most important features are "# of candidates" and "# of yes/no questions" and the least important features are "yes/no answer" and "different of top 1".

Table 2.7: The results of when to stop on the abstract level (top) and the document level (bottom) with different classifiers. The average number of asked questions is shown in parentheses following each total effort.

	MAP			Effort		
Stopping point	SVM	random forests	neural network	SVM	random forests	neural network
10%	0.765	0.651	0.491	351 (159)	526 (122)	724 (60)
15%	0.839	0.697	0.617	269 (125)	481 (97)	601 (52)
20%	0.814	0.758	0.550	174 (95)	489 (81)	513 (39)
25%	0.871	0.812	0.696	96 (69)	347 (39)	371 (24)
30%	0.900	0.746	0.643	74 (66)	221 (48)	214 (29)
35%	0.895	0.823	0.695	63 (58)	94 (34)	46 (27)
40%	0.900	0.807	0.679	82 (77)	77 (71)	54 (34)
45%	0.805	0.832	0.756	181 (45)	46 (39)	46 (29)
50%	0.938	0.857	0.753	46 (44)	26 (21)	27 (16)
55%	0.917	0.988	0.858	17 (16)	17 (16)	16 (11)
60%	0.917	0.917	0.550	14 (12)	14 (12)	15 (9)
65%	1.000	1.000	0.100	13 (12)	13 (12)	17 (7)
70%	1.000	1.000	0.250	13 (12)	13 (12)	13 (9)
75%	1.000	1.000	1.000	12 (11)	11 (10)	11 (10)
80%	1.000	0.500	0.200	12 (11)	11 (9)	12 (7)
	MAP			Effort		
Stopping point	SVM	random forests	neural network	SVM	random forests	neural network
10%	0.825	0.771	0.768	68 (62)	83 (63)	85 (77)
15%	0.761	0.608	0.851	34 (30)	36 (31)	31 (28)
20%	0.742	0.750	0.693	16 (13)	16 (13)	16 (13)
25%	0.821	0.929	0.895	10 (8)	10 (9)	12 (10)
30%	0.833	0.833	0.770	9 (7)	9 (7)	12 (9)
35%	1.000	0.833	1.000	8 (7)	9 (7)	11 (10)
40%	0.583	0.583	1.000	12 (9)	12 (9)	13 (11)
45%	0.833	0.833	0.225	10 (7)	11 (8)	13 (5)
50%	0.500	0.500	0.500	8 (6)	7 (5)	8 (6)
55%	0.500	0.500	0.100	7 (5)	6 (4)	12 (2)

2.4.6 Online User Study

In addition to the simulated users, we also develop an online system and involve an information specialist,[9] whose job is to conduct searches in TAR with the assistance of medical experts, to conduct a small online user study to confirm some of the assumptions made in this chapter and evaluate how well our recommender system works "in-situ". The ideal users would be medical experts, who are conducting a TAR to write a

[9]The information specialist is an internationally renowned expert in the domain of evidence synthesis with more than 10 years work expertise, a member of the Council of the international Cochrane Collaboration, and a member of the International Collaboration of the automation of systematic reviews (ICASR).

Table 2.8: Relative feature importance on the abstract level (top) and the document level (bottom) by random forests.

Feature	Importance score
# of candidates	0.543490
# of yes/no questions	0.195774
Stopping point	0.112272
different of candidates	0.074370
different of preference	0.035941
yes/no answer	0.027093
different of top 1	0.011060

Feature	Importance score
# of candidates	0.487968
# of yes/no questions	0.246702
Stopping point	0.126864
different of candidates	0.075180
different of preference	0.036955
different of top 1	0.013730
yes/no answer	0.012601

review paper within their well-understood topic and have already found most of the relevant documents using a BMI-based platform, and now they converse with our system embedded in the platform to find their last few missing relevant documents. However, finding such an expert user base who are currently conducting a TAR is not feasible. Thus, in our study we asked the information specialist to choose all the studies he feels comfortable with out of the 50 topics in our collection. Then, we assume that the BMI has already run, and we provide the specialist with all the found relevant documents. The specialist is guided to review these relevant documents very carefully to familiarize himself with the topic. The missing relevant documents here are unknown to the specialist. After the specialist indicates that he is familiar with the topic, the conversation with the system can start. A question is selected by our algorithm to be asked to the specialist. The specialist is required to provide an answer for each question according to his expert knowledge and the topic information he read during the previous step, and then our system updates its relevance belief based on the provided answer. The specialist can stop answering questions any time during his interaction with the system. When stopping the interaction with the system, he is asked a number of exit questions about his experience with the system. Note that this is by no means an optimal in-situ user study.

During our user study the information specialist decided to work on 7 topics, i.e., 7 systematic reviews. We collected 193 rounds of questions generated by our system for the information specialist on these 7 topics. First, we want to understand how many questions the user is willing to answer, and how well the user answers them. From the collected data, we observe that the specialist answered an average number of 28

Table 2.9: Two examples of a sequence of questions asked by our model.

Topic: Optical coherence tomography (OCT) for detection of macular oedema in patients with diabetic retinopathy		
Target documents: (missing relevant documents): ID: 15051203, Title: Comparison of the clinical diagnosis of diabetic macular edema with diagnosis by optical coherence tomography. ID: 9479300, Title: Topography of diabetic macular edema with optical coherence tomography.		
Question	**Answer**	**Rank of Last Relevant**
Are the documents about ...		188
diabetic macular edema (DME)	Yes	69
treatment	No	48
retina	Not Sure	48
optical coherence tomography (OCT)	Yes	27
measurements	Yes	17
evaluation	No	2

Topic: Human papillomavirus testing versus repeat cytology for triage of minor cytological cervical lesions		
Target documents (missing relevant documents): ID: 19116707, Title: Prevalence of human papillomavirus types 6, 11, 16 and 18 in young Austrian women - baseline data of a phase III vaccine trial. ID: 19331088, Title: Cervical cytology screening and management of abnormal cytology in adolescents.		
Question	**Answer**	**Rank of Last Relevant**
Are the documents about ...		988
Human Papillomavirus (HPV)	Yes	430
women	Not Sure	430
cervical cancer	Yes	224
infection	Yes	129
cancer	Yes	44
development	No	19
treatment	Not Sure	19
disease	Yes	6
clinic	No	5
cervical	Yes	2

questions per topic using our system. Further, in the exit questionnaire, the specialist declares that he is willing to answer 30 questions. In the exit questionnaire, he thinks the conversational system is helpful and he is willing to use it in the future. Last, the specialist provided the correct answers to the system's question 76% of the time, he

was not sure about the answer 4% of the time, and he gave the wrong answer (i.e., his answer disagreed with the description of the missing relevant documents) 20% of the time. As we can see from Figure 2.6, most of the results of our model are higher than the state-of-the-art baseline BMI when there are 20% of noisy answers.

In this chapter, we calculate the total effort as the sum of the rank of the last relevant document and the number of asked questions. Therefore, we make the assumption that answering a direct question about entities requires at most as much effort as providing the relevance of a document. We attempt to validate this assumption through the user study. We recorded the time the specialist spent on reviewing each relevant document (title and abstract) before answering questions and the time the specialist spent on answering each question. From the collected data, we observe that the specialist took an average time of 55.8 seconds to screen each relevant document while spending only 7.8 seconds to answer a question, on average. Additionally, in the exit questionnaire, the specialist indicates that the system's questions were easy to answer. This observation is in agreement with the conclusions made in previous work which say that when judging the relevance of documents it is more effective and efficient to judge a provided document summary [158] or even better to judge relevant sentences [212].

2.5 Conclusions and Discussion

In this chapter, we aim to achieve high recall in TAR. We describe our interactive method, SBSTAR, which directly queries reviewers whether an entity is present or absent in missing relevant documents [224]. The framework applies a CAL algorithm on the relevance feedback from reviewers until a stopping point, which is a certain percentage of documents that have been reviewed, and then switches to locate the last few missing relevant documents. In this chapter we present three rudimentary models to model reviewers' noisy answers, as well as the noise-tolerance algorithm in this chapter. We further conduct an analysis under different noisy answer simulation settings. We also propose an extension of SBSTAR, $\text{SBSTAR}_{\text{ext}}$, which performs a novel when-to-stop method by training a classifier to determine when to stop asking questions automatically, so as to avoid pre-setting the parameter of the number of questions when question asking. Additionally, we explore the performance of SBSTAR and $\text{SBSTAR}_{\text{ext}}$ at different levels, namely the abstract level and the document level. Experiments on the CLEF 2017 e-Health Lab dataset demonstrate SBSTAR can efficiently locate the missing relevant documents while asking for minimal reviewer effort. When accounting for noisy answers, the noise-tolerance algorithm is also effective in case reviewers are not able to provide 100% correct answers. Our experiment on the performance of $\text{SBSTAR}_{\text{ext}}$ demonstrates that our stopping to ask question model is effective. Last, we conduct a small user study that validates the availability of our assumptions about users' willingness and ability to answer a number of questions, as well as their efforts.

We introduce a question-based approach for the TAR task and validate the effectiveness of the question-based approach. However, this is just the first step towards an intelligent question-based system for assisting TAR and there is still room for improvement. A potential research direction is to incorporate the latest conversational / dialogue system techniques and question answering techniques to improve TAR, and

to aid it towards a perfect intelligent system. Furthermore, we ask questions about informative terms to locate the last few relevant documents after deploying the CAL algorithm which queries on documents. Instead of asking questions after deploying the CAL algorithm, one can also explore the switch mechanism for asking questions about informative terms and querying on documents in each iteration.

In this chapter we pivot around the presence or absence of entities in the target documents to generate questions to ask to the user. Recognizing entities in biomedical texts is a research direction of its own, with significant recent work on neural methods, which further advances the state of the art [76, 189]. In this chapter, we use TAGME, which is widely used in prior research, in semantic mapping [81, 198, 199], and biomedical information labeling [57, 137, 224]. However, this automatic entity annotation may provide some irrelevant annotations or may miss some entities in the data. Our method could certainly benefit from better entity recognition and salience detection methods, constructing more reliable and salient question pools. Similarly, there may be a richer set of possible questions to be asked, questions that may or may not be answered with a "yes" or a "no". For instance, questions could be constructed by using labeled topics [221, 223], keyword extraction [23], or other information extraction techniques [126, 149]. A richer type of questions could also be constructed by identifying properties of the documents (entities in a knowledge base triplet representation) and their relation to the document. For example, the following entities could be identified in the document description: "author", "year", "publisher", "subject category", patient population, intervention, comparison, outcome [187]. Questions then could be constructed from the derived triplets [148].

Further, our work simulates user answers, noisy or not. In the no-noise setting, we assume that when presented with an entity reviewers know whether the entity is present in all missing documents with 100% confidence. In the noisy setting, we propose some noise model to explore the performance of our algorithm and relax the aforementioned 100% confidence assumption. We define three noise settings in this chapter. However, these three settings are ad-hoc and the noise can also be defined as any function of any characteristic of entities, reviewers or topics. A large user study could be particularly helpful in understanding how and why users give erroneous answers to such system questions. Our simulation setup is just a first step towards considering noisy answers, something that is utterly missing from past work on the topic. Our small user study indicates that there is validity in the assumptions we have made, but yet again there is a need for an in-situ larger study to confirm that.

It is also likely that an entity may be semantically related to the desired document, while not lexically present. In this chapter we do not explore any semantic correlation modeling, but we leave it as future work.

Our when-to-stop algorithm $\text{SBSTAR}_{\text{ext}}$ is based on extracted dynamic features. In this chapter we extract seven related features to decide when to stop asking effectively. However, a systematic feature engineering investigation may discover more strong features related to automatically stop asking and feature selection algorithm may yield improvements, which we leave it as future work.

We apply GBS over entities to construct our objective function to find the best question to ask. Any other strategy learning techniques could also be used. We leave identifying more systematic objective functions as future work.

In this chapter, we focus on the task of TAR often performed in empirical medicine or legal e-discovery by expert reviewers. It is also possible to deploy our framework to other high-recall applications for naive users, but for that to be more effective we might need to change our TAR-specific prior belief initialization model BMI to the corresponding task-specific prior belief initialization models, and most likely accounting for other characteristics of the entities in comparison with the background level of the user.

Finally, in this chapter we interact with expert reviewers by asking questions to locate the last few relevant documents. Another possible way of locating these last few relevant documents is to keep reformulating the query. Query reformulation has shown its effectiveness to locate the targets for initial query mismatching and limited coverage [48]. However, users need to find the association between queries and incorporate the new information gained from the previous search by themselves to reformulate the next query. Furthermore, query reformulation may generate some duplicate results and reviewing them will cost extra effort. Our work automatically selects questions to ask and incorporates the answers to refine the search results, which can be a complement of keeping query reformulation. One can also combine our method with query expansion or reformulation techniques to a guided query expansion or reformulation.

In this chapter, to answer **RQ1**, we have provided a solution by proposing a question-based approach for the TAR task to assist document search and validate the effectiveness of asking CQs by the proposed question-based approach. Next, we focus on asking CQs to assist product search.

3 Question-based Product Search

In this chapter, we propose a novel question-based product search algorithm to effectively locate the best matching product, by asking clarifying questions (CQs). Like Chapter 2, the question-based product search algorithm queries users about the expected presence of an informative term (entity) in product related documents. In Chapter 2, we only learn a belief over document relevance. This is, we treat all CQs equally and ignore the differences of the informativeness and effectiveness of CQs. To that end, in this chapter, we perform a cross-user duet training to learn both a belief over product relevance and the reward over the performance of questions. We answer the following research question:

RQ2 How can we effectively ask CQs to improve product search?

3.1 Introduction

Purchasing goods and services over the Internet is by many considered a convenient and cost-effective shopping method [7]. This has led to an ever-increasing focus on online shopping technologies. The first and foremost stage of online shopping is generally recognized to be that of product search [155]. In product search, users usually formulate queries to express their needs and find products of interest by exploring the retrieved results.

When it comes to product search, the majority of methods use some similarity functions to match the query and documents that describes a product. In these cases, the query and product documents[1] are usually represented as vectors in observed or latent vector space and the distance between these vectors is computed [3, 72, 73, 180]. However, the queries in product search are often too general to capture the minute details of the exact product a user is looking for [52, 53].

Although the problem of efficiently finding the best match for a query with similarity and representation learning in a given set has been studied before [3, 72, 73, 180], interactive methods that can help users better specify their needs still remain underexplored. The main focus of this chapter is to effectively find the best matching product for the user by asking "yes"/"no" questions to the searchers. Given a pool of available

This chapter was published as [219].

[1] In this chapter, we use product documents to refer to product descriptions and reviews.

questions to be asked on products, for which the answer can either be "yes", or "no", or "not sure", our method performs a duet training, to learn both the product relevance and the informativeness of a question. That is, our proposed method uses limited data from existing users to construct a system belief over product relevance for any new user/query. Further, it is trained to select questions with the highest potential reward based on the system belief over product relevance. In particular, our method extends Generalized Binary Search (GBS) [131] over questions to find the question that best splits the probability mass of predicted product relevance and has the highest trained reward for the remaining of the products. After the question is being asked and answered by the user a posterior belief over product relevance is obtained and used for the selection of the next question. Lastly, the true belief over product relevance for the user is revealed sequentially through interactions with the user to generate the final product search results.

The main contribution of this chapter is three-fold:

- A novel interactive product search method based on constructed questions, QSBPS, which directly queries users about the expected presence of an informative term in product related documents.
- A method that learns question rewards and cross-user system beliefs with limited data.
- An extensive analysis of the performance of the algorithms that includes an analysis of noise tolerance in user answers.

The evaluation results show that our approach QSBPS can achieve better results measured by Recall@5, and Normalized Discounted Cumulative Gain (NDCG) compared to the state-of-the-art product search baselines.

3.2 Related Work

3.2.1 Product Search

E-shopping and e-retailing are attracting more and more attention, which has led to new developments in technology [155]. Most previous product search methods focus on structured information about products, e.g., brands, types and categories. Based on a semantically annotated product family ontology, Lim et al. [115] presented a multi-faceted product search and retrieval framework. Vandic et al. [181, 182] noticed that product information on pages is usually not well-structured, and proposed a faceted product search algorithm by using semantic web technology. Duan et al. [52, 53] observed that there is a vocabulary gap between product specifications and search queries, and developed a probabilistic mixture model to systematically mine product search logs by learning an attribute-level language model. Van Gysel et al. [180], on the other hand, introduced a latent vector space model to learn representations for products based on their associations with unstructured documents, which avoids the limitations of searching in structured data. They learned distributed representations of a given word sequence and products as well as a mapping between query and product space. Ai

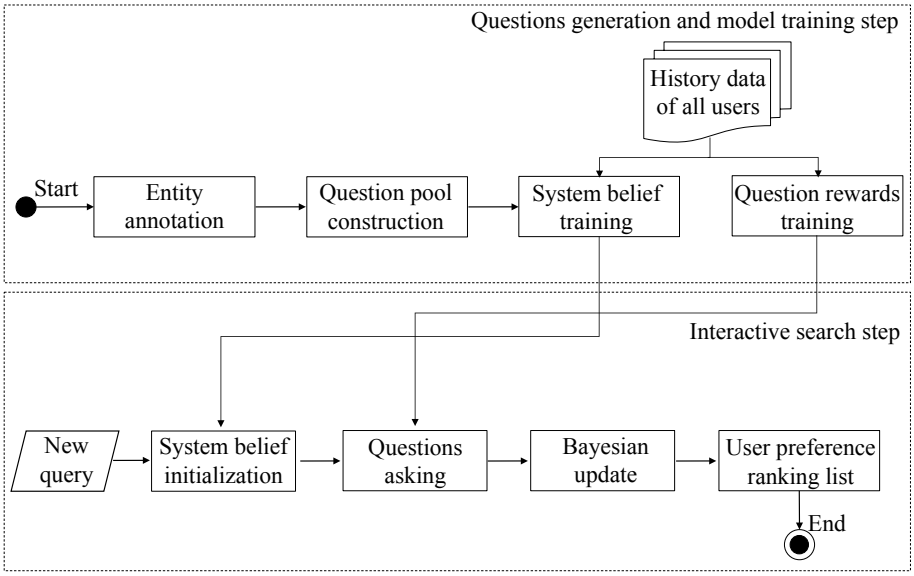
et al. [3] presented a hierarchical embedding model to learn semantic representations for queries, products, and users. They constructed a latent space retrieval model for personalized product search. Zamani and Croft [205] proposed a general framework that jointly models and optimizes a retrieval model and a recommendation model. Some studies also investigated the effectiveness of incorporating external information. Guo et al. [72] presented the translation-based search (TranSearch) model, which tries to match the user's target product from both textual and visual modalities by leveraging the "also_view" and "buy_after_viewing" of products. Then Guo et al. [73] proposed an Attentive Long Short-Term Preference model (ALSTP) for personalized product search by considering the long-term and short-term user preferences using two attention networks. Similar to previous work we also hypothesize that there is a gap between the language of product documents and user queries. Different from past work, we propose a question-based sequential Bayesian interactive learning from user preferences to learn their actual needs. Past work is complementary to ours, since it can be leveraged to inform a prior on product relevance, but it can also be used to construct questions on the basis of product structured information.

Zhang et al. [213] proposed a unified paradigm for product search and recommendation that constructs questions on aspect-value pairs, to ask the user questions over aspects. Their model obtains the opinion of the user (i.e., value of the aspect-value pair) for the "aspect" as feedback and thus expand the representation of the user query. Different from work that is based on aspect-value pairs and constructs the question by manual predefined language patterns, we ask questions about automatically extracted informative terms without complex language patterns. Further, the selection of questions by Zhang et al. [213] relies on the log-likelihood of probability estimation over limited aspects while our question selection is based on the cross-user duet learning of question effectiveness and user preferences.

3.2.2 Interactive Search

Quantifying relevance on the basis of users' queries, or learning a model of relevance from past queries, cannot always capture the minute details of relevance, not only in product search, but also in other search tasks, such as those that require high recall [35]. Interactive information retrieval, instead, suggests putting the human in the loop and learning a relevance model throughout an interactive search process, where users provide feedback on the relevance of presented documents, and the model adapts to this feedback [36]. Most of the methods for interactive search take a special treatment of the query [213], typically expanding it with terms from labeled documents. However, query expansion has show suboptimal performance [35], in part because handling the relation between the original query and feedback documents is challenging [123]. Active learning [36] and multi-armed bandits [85] have also been proposed to iteratively learn task-specific models. Different from the afore-mentioned methods that focus on receiving feedback at the level of documents, our interactive method asks explicit questions to the users in terms of entities contained in the documents of the collection. Similar to our work, Wen et al. [193] proposed a sequential Bayesian search (SBS) algorithm for solving the problem of efficiently asking questions in an interactive search setup. They learn a policy that finds items in a collection using the minimum number

Figure 3.1: Research framework



of queries. Based on Wen et al. [193], Zou et al. [224] devised the SBSTAR algorithm to find the last few missing relevant documents in Technology Assisted Reviews by asking "yes" or "no" questions to reviewers. Our QSBPS algorithm presented in this chapter differs by performing a cross-user duet training, to learn not only a belief over product relevance but also the reward over the performance of questions, as well as their noise-tolerance.

3.2.3 Learning to Ask

Learning to Ask is a new field of study that has recently attracted considerable attention. A number of studies focus on identifying good questions to be asked in a 20 Questions game setup. Chen et al. [28] presented a Learning to Ask framework, within which the agent learns smart questioning strategies for both information seeking and knowledge acquisition. Hu et al. [87] proposed a policy-based reinforcement learning method also within a 20 Questions game setup, which enables the questioning agent to learn the optimal policy of question selection through continuous interactions with the users. Both aforementioned works introduce data-hungry techniques, which require having large numbers of repeated interactions between users and the information seeking system to train upon. Different from these approaches, our method does not require having multiple searchers and their interactions for a given product.

3.3 Methodology

In this section, we provide a detailed description of the proposed method.[2] The research framework is shown in Figure 3.1. Our approach consists of three parts: (a) the construction of a pool of questions; (b) system belief and the question reward training using historical data from each user individually, and across users; and (c) an interactive search step, which sequentially selects questions to be asked to the user and updates the user preferences. The focus of this chapter lies with the two latter parts.

3.3.1 Question Pool Construction

The proposed method of learning informative questions to ask to users, described in detail in Sections 3.3.2 and 3.3.3, depends on the availability of a pool of questions about product properties. That is, given a product, the user should be able to answer the question, with a reference to the relevant product, with a "yes" or a "no".

There are different methods one could employ to construct such a set of questions. For instance, one can use labelled topics [222, 223], extracted keywords [74], item categories and attributes, or extract triplets and generate a rich set of questions based on these triplets [148]. In this chapter, we take a rudimentary approach. Our assumption is that a user can discriminate between products based on the language of the documents (i.e., descriptions and reviews) they are associated with. We then identify informative terms (instantiated by entities in this chapter) using the entity linking algorithm TAGME [61], similar to previous research [198, 199]. We assume that these informative terms comprise the most important characteristics of a product, and we generate questions about the presence or absence of such entities in product related documents. That is, we instantiate the question candidate set by identifying entities in the product related documents. Consider, for example, the following description of a product, "Apple iPhone XS (64GB), gold color. The most durable glass ever in a smartphone. And a breakthrough dual-camera system. iPhone XS is everything you love about iPhone." the extracted entities can be "Apple", "iPhone XS", "gold color", "smartphone", "dual-camera system", and "iPhone". We don't use any filter on the annotation scores of the TAGME results, i.e., all annotations are being considered, which is also a widely used setting in previous work [147, 198]. After that, the proposed algorithm asks a sequence of questions of the form "Are you interested in [entity]?" to locate the target product.

3.3.2 Cross-user Duet Learning

In this section, we describe the training algorithm that jointly learns (a) a belief over the effectiveness of questions (entities) in identifying relevant products, $R_t(e)$, and (b) a system belief over the relevance of the products, $\mathbb{P}_l(\pi)$. Instead of using the SBS algorithm [193], a data-hungry algorithm that requires a large amount of training data for each user's request, our approach uses a duet learning approach on the given topic, t.[3] The proposed method updates the system belief over relevance and the entity

[2] Source code: https://github.com/UvA-HuMIL/QSBPS

[3] Topics in this chapter are product subcategories, which can represent user queries.

Algorithm 3: QSBPS Offline Learning

input : A product documents collection, $\mathcal{D}$, the set of annotated entities in $\mathcal{D}$, $\mathcal{E}$, a set of topics $\mathcal{T}$, a prior Dirichlet parameter, $\alpha_{\mathbf{0}}$

output : System belief $\mathbb{P}_t(\pi)$, question rewards $R_t(e)$

foreach *topic* $t \in \mathcal{T}$ **do**
  Compute the initial user preference with $\alpha_{\mathbf{0}}$:
  $\pi_0^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_0}[\pi(d)] \quad \forall d \in \mathcal{D}$
  Let $\mathcal{D}_t$ be the set of products within t
  $n \leftarrow 0$
  foreach $d \in \mathcal{D}_t$ **do**
    foreach *entity* $e \in \mathcal{E}$ **do**
      Update the system's belief $\mathbb{P}$ using Bayes' rule:
      $\mathbb{P}_{n+1}(\pi) \propto \pi(d)\mathbb{P}_n(\pi) \quad \forall \pi$
      Calculate reward for each entity: $R_d(e) = \frac{I_{before} - I_{after}}{|U|}$
      $n \leftarrow n + 1$
    end
  end
  Output trained system belief $\mathbb{P}_t(\pi) = \mathbb{P}_{n+1}(\pi)$
  Output average reward for each entity: $R_t(e) = \frac{1}{|\mathcal{D}_t|} \sum_{d \in \mathcal{D}_t} R_d(e)$
end

effectiveness after every question is being answered by the user, performing well using limited and weak signals. The system belief training over products learns the interest of the users over products, while the entity effectiveness training over entities learns the reward or informativeness of questions, and thus finds the optimal policy for asking questions. Our algorithm performs two rounds. During the offline phase, the algorithm learns what is the average users' preference over the products within each product category, and how effective entities are in identifying these products. Then, during the online interactive search phase, the algorithm continues learning product relevance on the basis of the user's answers to the algorithm's questions.

The offline training phase is described in Algorithm 3. We assume that there is a target relevant product $d^* \in \mathcal{D}$.[4] The user preferences for the products are modelled by a (multinomial) probability distribution π^* over products $\mathcal{D}$, and the target product is drawn i.i.d. from this distribution. We also assume that there is a prior belief $\mathbb{P}_0$ over the user preferences π^*, which is a probability density function over all the possible realizations of π^*. The prior system belief $\mathbb{P}_0$ is a Dirichlet distribution, with a hyper-parameter α_0, which can be set by using any other product search algorithm that measures the lexical or semantic similarity between the query and product documents, or any collaborative filtering method. In this chapter, we use an uninformative uniform system belief distribution by setting all α_0's to 1 to isolate the effect of the proposed method. That is, the user preference is initialized to be the same for each product.

During the duet training phase, there is a training set D_t for each topic, which

[4] In the rest of chapter, "product" and "product document" will be used interchangeably.

Algorithm 4: QSBPS Online Learning

input : A product documents collection, $\mathcal{D}$, the set of annotated entities in $\mathcal{D}$, $\mathcal{E}$, a set of topics $\mathcal{T}$, a number of questions to be asked, N_q, the system belief $\mathbb{P}_t(\pi)\ \forall t \in \mathcal{T}$, and the question rewards $R_t(e)\ \forall t \in \mathcal{T}$.

output : User preference $\pi^*_{N_q}$

foreach *topic* $t \in \mathcal{T}$ **do**
- Load $\mathbb{P}_t(\pi)$, $R_t(e)$
- $l \leftarrow 1$
- System belief initialization: $\mathbb{P}_l(\pi) \leftarrow \mathbb{P}_t(\pi)$
- **while** $l \leq N_q$ *and* $|U_l| > 1$ **do**
  - Compute the user preference with $\mathbb{P}_l(\pi)$: $\pi^*_l(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] \quad \forall d \in \mathcal{D}$
  - Find the optimal target entity: $e_l = \arg\min_e |\sum_{d \in u_l} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d)| - \gamma * R_t(e)$
  - Ask the question about e_l, observe the reply $e_l(d^*)$
  - Remove e_l from the entity pool
  - $U_{l+1} = U_l \cap \{i \in \mathcal{D} : e_l(i) = e_l(d^*)\}$
  - Update the system's belief: $\mathbb{P}_{l+1}(\pi) \propto \pi(d)\mathbb{P}_l(\pi) \quad \forall \pi$
  - $l \leftarrow l + 1$
- **end**

end

contains all of the training products for this topic. For each product in D_t, we generate a set of questions based on all the entities in the collection, and we obtain a posterior belief using the Bayes' rule after every question for the target training product d is being answered, and calculate the reward $R(e)$ for each question (entity). We then get the average reward for each entity to be used in the online interactive search, and obtain the trained system belief $\mathbb{P}_t(\pi)$, which is also used as a prior belief over products during the interactive search.

System Belief: We learn the system belief from the training data of all users on a certain topic, assuming that the training products on a certain topic are related in the entity embedding space, and thus can provide useful guidance. One could also learn user personalized preferences and entity informativeness, however, we do not do so, hypothesizing that users can buy significantly different products.

Let $\mathbb{P}_l$ be the system's belief over π^* in the n-th question. We compute the user preference $\pi^*_l(d)$ in the n-th question by:

$$\pi^*_n(d) = \mathbb{E}_{\pi \sim \mathbb{P}_n}[\pi(d)] \quad \forall d \in \mathcal{D}. \tag{3.1}$$

Similar to Wen et al. [193] and Zou et al. [224], we model the user preference π^* by a multinomial distribution over products $\mathcal{D}$. Then, the system updates its belief after observing the user answer to a question asked, which is sampled i.i.d. from π^*:

$$\mathbb{P}_{n+1}(\pi) \propto \pi(d)\mathbb{P}_n(\pi) \quad \forall \pi. \tag{3.2}$$

We model the prior $\mathbb{P}_0$, by the conjugate prior of the multinomial distribution, i.e., the Dirichlet distribution, with parameter α. Further, we define the indicator vector $Z_l(d) = \mathbb{1}\{e_l(d) = e_l(d^*)\}$, where $e_l(d)$ means whether the product d contains entity e_l or not. From Bayes' rule, the posterior belief prior to the question l is:

$$\mathbb{P}_{n+1} = Dir\left(\alpha + \sum_{j=0}^{n} Z_j\right). \tag{3.3}$$

From the properties of the Dirichlet distribution, we have:

$$\pi_n^*(d) = \mathbb{E}_{\pi\sim\mathbb{P}_n}[\pi(d)] = \frac{\alpha(d) + \sum_{j=0}^{n} Z_j(d)}{\sum_{d'\in\mathcal{D}}(\alpha(d') + \sum_{j=0}^{n} Z_j(d'))}, \tag{3.4}$$

where $\alpha(d)$ is the i-th entry of α, which corresponds to product d. Therefore, the user preference π_l^* can be updated by counting and re-normalization.

Question Reward: For question reward learning, we use historical training data to learn the reward of each entity. We define the following simple reward function, which can learn the ranking rising ratio of the target product relative to the candidate products version space when training:

$$R(e) = \frac{I_{\text{before}} - I_{\text{after}}}{|U|}, \tag{3.5}$$

where I_{before} is the index of the target product in the ranked list before asking the question about entity e, I_{after} is the index of target product in the ranked list after asking the question about entity e, and $|U|$ is the number of products in the candidate set U. The ranked list is generated according to the user preference π_l^* over the products. Note that we use the worst ranking index as the index of product in the ranked list when there are ties over the user preferences. Thus, in the first question, I_{before} is initialized to the last ranking index, which is equal to the number of products in the collection.

After the system belief learning and question reward learning, the model uses its current user preference $\pi^*(d)$ from the belief $\mathbb{P}_t$ and the estimated reward $R_t(e)$ to derive a policy to find the optimal entity to query.

3.3.3 QSBPS Algorithm

In this section, we introduce two versions of the QSBPS algorithm. The first version assumes that there is no noise in the answers of a user. That is, when an entity appears in the text of the relevant product the user gives a correct positive answer, while when it does not the user gives a correct negative answer (see user simulation in Section 3.4.1). The online interactive learning of our proposed algorithm is provided in Algorithm 4. We first load in the trained system belief $\mathbb{P}_t(\pi)$ and question rewards $R_t(e)$, then compute the user preference with prior belief equal to $\mathbb{P}_t(\pi)$, and find the optimal entity e_l by Equation 3.6. Inspired by Wen et al. [193] and Zou et al. [224], we extend GBS over entities to find the entity that best splits the probability mass of predicted product

relevance closest to two halves, but also maximize the question reward for the remaining of the products during l-th question, as the optimal entity:

$$e_l = \arg\min_e \left| \sum_{d \in u_l} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d) \right| - \gamma * R(e), \tag{3.6}$$

where e_l is the l-th chosen entity, u_l is the set of products of the candidate version space when asking the l-th question, $e(d)$ expresses whether the product d contains the entity e or not, while γ is the weight to trade the question reward $R(e)$. We ask whether the entity e_l is present in the target product that the user wants to find, d^*, observe the reply $e_l(d^*)$, and remove e_l from the entity pool. Then we reduce U_l, update the system's belief $\mathbb{P}_l$ using Bayes' rule and recalculate the user preference, i.e., the user preference is updated sequentially by Bayesian learning that refers to sequential Bayesian based search. Since the user preference π^* is a multinomial distribution over products $\mathcal{D}$, and $\mathbb{P}_t$, a Dirichlet distribution, updating the system belief is performed in a similar to Equation 3.3 manner.

In Algorithm 4, we make the assumption that users, when presented with an entity, know with 100% confidence whether the entity appears in the target product. To relax this assumption we also propose a noise-tolerant version of the algorithm. That is, we allow the user to make mistakes and provide the algorithm with the wrong answer. We integrate the probability that the user will give the wrong answer to a question about entity e, $h(e)$, into the new objective function, at line 7 of Algorithm 4:

$$e_l = \arg\min_e \left| \sum_{d \in \mathcal{D}} (2\mathbb{1}\{e(d) = 1\} - 1)\pi^*(d) \right| + 2\beta * h(e) - \gamma * R_l. \tag{3.7}$$

We observe the noisy answer and update the posterior system belief according to this noisy answer. Intuitively, a question will be chosen to be asked not only if it is about an informative entity, but also if this entity is the one for which users have a good confidence in providing an answer. The experiments will be discussed in **RQ2.4**. Regarding the error rate for each entity $h(e)$, we consider two settings: In the first setting all of the $h(e)$ are simply set to equal values, and we experiment with different error rates that range from 0.1 to 0.5 with a step of 0.1, to explore the performance trend of our model with different error rates. An error rate $h(e)$ of 0.5 means that the user has a 50% probability to give the wrong answer. In the second setting, given that users are usually more confident in their answers about an entity e if e is frequently occurring in the given topic, we define $h(e)$ as a function of average term frequency (TF) in the topic for each entity, which is in the range of (0, 0.5]:

$$h(e) = \frac{1}{2(1 + \text{TF}_{\text{avg}}(e))}, \tag{3.8}$$

where $\text{TF}_{\text{avg}}(e)$ represents the average term frequency of entity e in the given topic. The choice of this function is ad-hoc and any other function of any other characteristic of entities could also be used. Ideally, one should conduct a user study to identify a reasonable error rate function, the properties of entities that affect the error rate, or even the characteristics of the users that influence the error rate. We leave such a user study as future work.

3.4 Experiments and Analysis

Through the experiments conducted in this chapter we decompose **RQ2** into the five following subquestions and aim to answer them:

RQ2.1 What is the impact of the number of questions asked and the parameter γ that trades the weight of question reward?

RQ2.2 What is the influence of using user reviews along with product descriptions?

RQ2.3 Does our duet training by using other users' data help?

RQ2.4 What is the performance when considering noisy answers?

RQ2.5 How effective is our proposed method for finding the best matching product compared to prior works?

3.4.1 Experimental Setup

Dataset. In our experiments we use the collection of Amazon products[5] [127]. It includes millions of customers, products and reviews. Each product contains rich metadata such as title, product descriptions, product categories, and also reviews from customers. Similar to Van Gysel et al. [180], we use the two product domains from the Amazon product dataset, which are "Home & Kitchen", and the "Clothing, Shoes & Jewelry". Statistics on these two domains are shown in Table 3.1. The documents associated with every product consist of the product description and the reviews provided by Amazon customers. To construct topics (queries) we use the method employed in Van Gysel et al. [180], and Ai et al. [3], i.e., we use a subcategory title from the above two product domains to form a topic string. Each topic (i.e., subcategory) contains multiple relevant products and products can be relevant to multiple topics. After that, we remove the topics which contain just a single product, since having a single relevant product does not allow constructing a training set and test set. Similar to [3], we randomly split the dataset to 70% and 30% subsets, and we use 70% of the products for each topic (i.e., subcategory) for training. We also use a 10% of the data as validation set. The validation set is used to select the optimal parameters to avoid overfitting.

Evaluation Measures. To quantify the quality of algorithms, we use Mean Reciprocal Rank (MRR), and average Recall@k (k= 5) and NDCG as the evaluation measures. We evaluate the performance of the algorithms for each individual user purchase observed in the data. Hence, for the same query but two different users, the relevant product (i.e., the product purchased) can be different. Therefore, in our dataset there is only a single relevant product for each query that resulted in a purchase, hence the use of MRR. Recall at rank 5, expresses whether the relevant document appears in the top-5 ranked products, while NDCG penalizes the effectiveness score by the rank at which the relevant product appears in a smoother way compared to MRR.

User Simulation. Our experimentation depends on users responding to questions asked by our method. Conducting a user study is in our future plans, however, in this chapter we follow recent work that simulates users [170, 213]. We simulate users

[5] http://jmcauley.ucsd.edu/data/amazon/

Table 3.1: Overview of the dataset. M denotes Metadata only and M&R denotes Metadata & Reviews. Arithmetic mean and standard deviation are indicated wherever applicable.

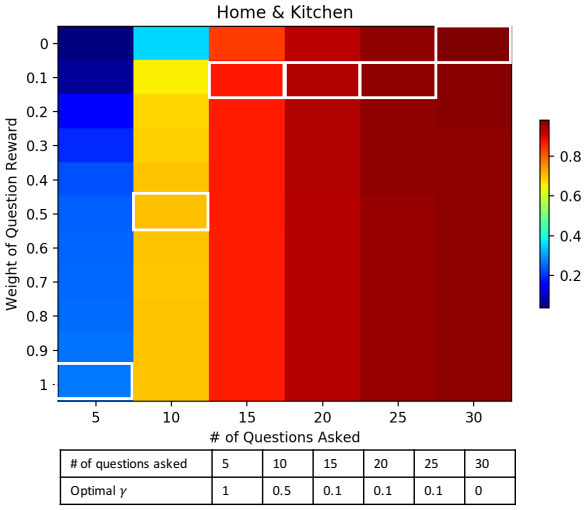
	Home & Kitchen	Clothing, Shoes & Jewelry
Number of products	8,192	16,384
Number of description docs	8,192	16,384
Number of reviews	79,938	77,640
Length of documents	70.02 ± 73.82	58.41 ± 61.90
Reviews per product	9.76 ± 52.01	4.74 ± 18.60
Number of topics	729	833
Products per topic	11.24 ± 31.16	19.67 ± 55.24
Number of entities (M)	232,086	385,727
Number of entities (M&R)	1,483,659	1,408,828
Entities per product (M)	28.33 ± 23.93	23.54 ± 19.25
Entities per product (M&R)	181.11 ± 797.55	85.99 ± 276.30

following two different settings: (1) we assume that the user will respond to the questions with full knowledge of whether an entity is present or not in the target product. Hence, we assume that a user has a product in mind, which is deterministic but unknown, and the user will respond with "yes" if an entity is contained in the target product documents and "no" if an entity is absent from the target product documents on the offline training phase and the online interactive search phase. This setting is the same used by Zhang et al. [213], which assumes that the user has perfect knowledge of the value of an aspect for the target product; (2) additionally, we allow the users to give the wrong answer to our product search system with a given probability during online interactive search. We consider two noisy answers settings, which are described in Section 3.3.3, while the precise experiment is described in **RQ2.4**.

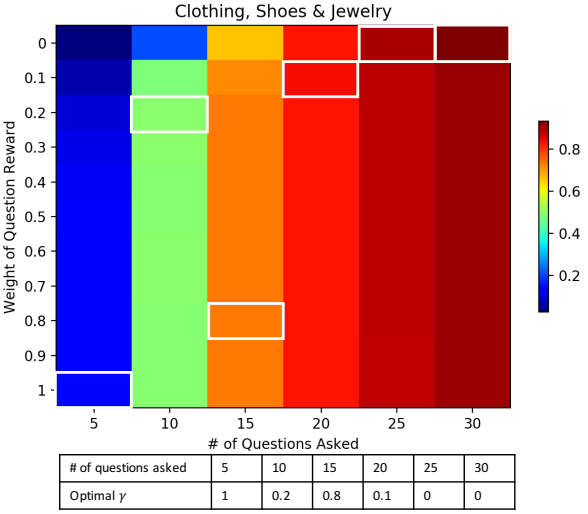
Baselines. We compare our method, called QSBPS, with six baselines. The first three baselines are interactive baselines while the last three ones are query-product semantic matching baselines: (1) **Random**, which randomly chooses the entity from the entity pool to ask a question about; (2) **SBS** [193], which is a sequential Bayesian search algorithm, that uses different training than our algorithm; (3) **PMMN** [213], i.e., the Personalized Multi-Memory Network, which is a state-of-the-art conversational recommender system asking questions on aspect-value pairs. Similar to the experiments run in the original paper, we assume that the system is able to retrieve the right candidate aspects for the product with 100% accuracy, which leads to an upper bound performance impossible to be actually reached by a real system; (4) **LSE**[6] [180], which is a state-of-the-art latent vector model, that jointly learns the representations of words, products and the relationship between them in product search; (5) **TranSearch**[7] [72], which is a state-of-the-art product search model using multi-modal preference modeling from both textual and visual modalities. For fair consideration, we use the textual version

[6]https://github.com/cvangysel/SERT
[7]https://github.com/guoyang9/TranSearch

Figure 3.2: Heatmap of the MRR results on the validation set of "Home & Kitchen" (top), and "Clothing, Shoes & Jewelry" (bottom). The MRR is shown as a function of the number of questions asked and the weight of question reward γ. The more red the heatmap, the better the performance of the method. The optimal weight parameter γ for the corresponding number of questions asked is designated by the white boundary box and is reported in the table below the heatmap. (Unless mentioned otherwise, in what follows the optimal weight parameter γ we used for the corresponding number of questions asked is designated in the figure.)



# of questions asked	5	10	15	20	25	30
Optimal γ	1	0.5	0.1	0.1	0.1	0



# of questions asked	5	10	15	20	25	30
Optimal γ	1	0.2	0.8	0.1	0	0

of their TranSearch model, i.e., TranSearch$_t$ with pre-training; and (6) **ALSTP**[8] [73], which is one of the state-of-the-art product search models using attention networks of long-term and short-term user preferences. For the latter four baselines, we use the

[8]https://github.com/guoyang9/ALSTP

optimal parameters reported in the corresponding product search papers.

3.4.2 **RQ2.1** The Impact of the Number of Questions and the Question Reward Parameter γ

In this section we answer **RQ2.1**. Our proposed method is parameterized by the number of questions to be asked to the user and the hyper-parameter γ which is the weight of the question reward in QSBPS. We evaluate the performance on the validation set. The number of questions asked to the user ranges from 5 to 30 with an interval of 5, and the weight γ ranges from 0 to 1 with an interval of 0.1. Figure 3.2 shows the heat map of MRR results for every combination of the number of asked questions and the question training controlling parameter γ on "Home & Kitchen" and "Clothing, Shoes & Jewelry" categories. The x-axis is the number of questions asked, and the y-axis are different values of the weight parameter γ. From Figure 3.2 it can be seen that the MRR increases with the number of questions asked to the user, as expected: the more the questions asked the better the performance of the algorithm. It can also be observed that MRR fluctuates over different question reward controlling parameter γ. Different numbers of asked questions produce different optimal values for γ. The optimal γ for 5, 10, 15, 20, 25, and 30 questions is 1, 0.5, 0.1, 0.1, 0.1, and 0, respectively on "Home & Kitchen", while 1, 0.2, 0.8, 0.1, 0, and 0, respectively, on "Clothing, Shoes & Jewelry". As we can see, the overall trend of the optimal value of γ is decreasing with the number of questions. This suggests that question reward training is more important in the early stages of question asking, especially when the number of questions is 5 and 10. When the number of questions asked is large, the importance of question reward training decreases. The reason could be that a large number of questions are sufficient for high performance regardless of reward.

3.4.3 **RQ2.2** The Influence of Using User Reviews Data

For **RQ2.2**, we explore the impact of using the user reviews data. We compare the differences between the results using product meta data only and the results using both product meta data and user reviews data. The results are shown in Table 3.2 with three metrics, MRR, Recall@5 and NDCG on the two product categories. For each number of questions asked, the near-optimal value of γ was used as indicated by the white-boundary boxes and tables in Figure 3.2. As shown in Table 3.2, we can see that three metrics are higher for the combined meta data and reviews data compared to only using meta data, especially when the number of questions is greater than 5. This indicates that user reviews are important in the users' buying process, and offer entities that can be more discriminative than the ones included in the product description documents only.

3.4.4 **RQ2.3** The Influence of the Duet Training

To answer **RQ2.3**, we investigate the impact of our duet training using other users' data, i.e., learning both the questions performance over entity and the system belief over products, by comparing it to (1) using no training, (2) using only questions performance

Table 3.2: A comparison of the MRR, and Recall@5, and NDCG results on "Home & Kitchen" (top), and "Clothing, Shoes & Jewelry" (bottom) by using metadata only versus using metadata & reviews. As it can be observed, user reviews, while noisy, can improve the selection of informative questions, by discussing interesting properties/entities of the products not present in the product descriptions.

	MRR		Recall@5		NDCG	
# of questions asked	Meta only	Meta & review	Meta only	Meta & review	Meta only	Meta & review
5	0.290	0.292	0.427	0.439	0.411	0.423
10	0.568	0.684	0.690	0.809	0.647	0.749
15	0.702	0.860	0.790	0.923	0.758	0.890
20	0.779	0.932	0.855	0.965	0.821	0.947
25	0.822	0.956	0.890	0.977	0.856	0.966
30	0.846	0.982	0.862	0.984	0.870	0.985

	MRR		Recall@5		NDCG	
# of questions asked	Meta only	Meta & review	Meta only	Meta & review	Meta only	Meta & review
5	0.140	0.145	0.213	0.213	0.274	0.288
10	0.327	0.486	0.462	0.645	0.445	0.588
15	0.490	0.719	0.630	0.835	0.585	0.777
20	0.575	0.819	0.715	0.906	0.657	0.859
25	0.668	0.897	0.736	0.928	0.724	0.917
30	0.745	0.930	0.802	0.960	0.789	0.945

Table 3.3: The performance of QSBPS when excluding all training (No-train), when including only questions effectiveness training (Q-train), when including only product relevance training (P-train), and performing the proposed duet training by using other users' data, on the "Home & Kitchen" (top) and "Clothing, Shoes & Jewelry" (bottom). As it can be observed duet training outperforms all other options demonstrating the suitability of the proposed method.

	MRR				Recall @ 5				NDCG			
# of questions asked	No-train	Q-train	P-train	Duet	No-train	Q-train	P-train	Duet	No-train	Q-train	P-train	Duet
5	0.00	0.14	0.04	0.29	0	0.21	0.05	0.44	0.12	0.29	0.18	0.42
10	0.13	0.53	0.35	0.68	0.27	0.69	0.59	0.81	0.30	0.63	0.49	0.75
15	0.78	0.77	0.84	0.86	0.82	0.85	0.89	0.92	0.82	0.81	0.87	0.89
20	0.91	0.87	0.93	0.93	0.92	0.92	0.95	0.97	0.93	0.89	0.95	0.95
25	0.95	0.91	0.96	0.96	0.96	0.94	0.97	0.98	0.96	0.93	0.97	0.97
30	0.98	0.92	0.98	0.98	0.98	0.95	0.98	0.98	0.98	0.94	0.99	0.99
	MRR				Recall @ 5				NDCG			
# of questions asked	No-train	Q-train	P-train	Duet	No-train	Q-train	P-train	Duet	No-train	Q-train	P-train	Duet
5	0.00	0.05	0.03	0.15	0	0.07	0.04	0.21	0.11	0.19	0.16	0.29
10	0.06	0.33	0.21	0.49	0.07	0.48	0.34	0.65	0.21	0.45	0.35	0.59
15	0.57	0.60	0.66	0.72	0.65	0.73	0.74	0.84	0.65	0.68	0.72	0.78
20	0.79	0.73	0.82	0.82	0.83	0.83	0.88	0.91	0.83	0.79	0.86	0.86
25	0.88	0.81	0.90	0.90	0.90	0.89	0.93	0.93	0.90	0.85	0.92	0.92
30	0.92	0.85	0.93	0.93	0.94	0.92	0.96	0.96	0.93	0.88	0.94	0.94

training on entities, and (3) using only system belief training on products. Our hypothesis is that the historical data of a specific user together with that of other users captures important information and the model training of our duet learning framework from these data will benefit even if the training data does not exactly match the information of the specific target user. Indeed, Table 3.3 shows the performance measured by the three metrics using duet training are higher than the ones that corresponds to excluding questions training, excluding system belief training or excluding both, especially when the number of questions is less than 20. For each number of questions asked, the near-optimal value of γ was used as indicated by the white-boundary boxes and tables in Figure 3.2. We conclude that using our duet learning framework is highly beneficial. When a large number of questions (greater than 15) is asked, the "No-train" achieves a higher performance than the "Q-train". This might be because "Q-train" only uses the noisy rewards trained by the weak signals and does not use the GBS policy like "No-train", "P-train", and "Duet" to select the optimal question to ask, and thus is less effective when the number of questions increases.

3.4.5 **RQ2.4** The Influence of Noisy Answers

Given that the user may not always give us the right answer, we also explore a noise-tolerance version of the QSBPS algorithm so as to answer **RQ2.4**. We develop a noise-tolerant version of the QSBPS algorithm as explained in Section 3.3.3 and investigate what the influence of noisy answers is. We simulate noisy answers of the user under two settings. In the first setting, we fix the probability of error, ε, and consider it as a parameter that ranges from 10% to 50% with step size 10%. The results when $h(e)$ is a fixed number ε are shown in Table 3.4. For each number of questions asked, the near-optimal value of γ was used as indicated by the white-boundary boxes and tables in Figure 3.2. It is obvious and expected that the performance as captured by the three metrics decreases with the increase of ε. When ε is equal to 50%, which means the user answer the question with "yes" or "no" at random, we observe very low performance. On the other hand, when ε is equal to 10%, the values of the three metrics are still relatively high. Further, note that, in comparison to the best query-product matching baselines, the TranSearch and ALSTP models, which are presented in Table 3.6, for 10% wrong answers, our method outperforms the baseline after 3 questions asked, and for 20% wrong answers, after 6 questions asked.

In the second setup, we define $h(e)$ as a function of term frequency of entity as shown in Equation 3.8. Similar to **RQ2.1**, we evaluate the MRR on validation set for the two categories, to select the optimal β. We report the optimal β here: the optimal β is 1 on "Home & Kitchen", and 0.8 on "Clothing, Shoes & Jewelry". The results when $h(e)$ is modelled by term frequency using the optimal weight parameter β are shown in Table 3.5. For each number of questions asked, the near-optimal value of γ was used as indicated by the white-boundary boxes and tables in Figure 3.2. As one can observe, the performance as measured by the three metrics when using the optimal β is higher than when β=0, which suggests that the objective function of our noise-tolerant version of QSBPS algorithm is effective. Further, note that our method, when the optimal β is being used, outperforms the best query-product matching baselines, the TranSearch and ALSTP models, presented in Table 3.6, after 4 questions have been asked, despite the

Table 3.4: The results of noisy answers on "Home & Kitchen" (top), and "Clothing, Shoes & Jewelry" (bottom) when $h(e)$ is a fixed, ranging from10% to 50% with a step 10%. Our method outperforms the best query-product matching baselines, the TranSearch and ALSTP model, which are presented in Table 3.6, after about 3 questions asked with 10% of wrong answers, and after about 6 questions being asked with 20% of wrong answers. Naturally the performance of our method is equal to random ranking when wrong answers are provided 50% of the time.

	MRR						Recall @ 5						NDCG					
# of questions asked	No-noise	10%	20%	30%	40%	50%	No-noise	10%	20%	30%	40%	50%	No-noise	10%	20%	30%	40%	50%
5	0.292	0.186	0.111	0.063	0.033	0.016	0.439	0.274	0.166	0.088	0.049	0.024	0.423	0.313	0.232	0.175	0.135	0.111
10	0.684	0.398	0.194	0.082	0.033	0.012	0.809	0.507	0.268	0.117	0.045	0.016	0.749	0.501	0.312	0.196	0.138	0.107
15	0.860	0.538	0.283	0.114	0.033	0.007	0.923	0.640	0.373	0.152	0.044	0.008	0.890	0.622	0.396	0.229	0.138	0.100
20	0.932	0.651	0.342	0.130	0.036	0.007	0.965	0.752	0.433	0.169	0.050	0.010	0.947	0.718	0.450	0.247	0.142	0.099

	MRR						Recall @ 5						NDCG					
# of questions asked	No-noise	10%	20%	30%	40%	50%	No-noise	10%	20%	30%	40%	50%	No-noise	10%	20%	30%	40%	50%
5	0.145	0.090	0.052	0.029	0.016	0.008	0.213	0.129	0.074	0.040	0.022	0.010	0.288	0.219	0.168	0.135	0.110	0.095
10	0.486	0.238	0.110	0.047	0.018	0.007	0.645	0.325	0.157	0.068	0.023	0.007	0.588	0.361	0.230	0.156	0.113	0.092
15	0.719	0.393	0.175	0.064	0.019	0.005	0.835	0.507	0.231	0.088	0.025	0.007	0.777	0.498	0.294	0.176	0.116	0.090
20	0.819	0.505	0.230	0.082	0.018	0.004	0.906	0.616	0.312	0.114	0.025	0.005	0.859	0.595	0.349	0.194	0.116	0.089

Table 3.5: The results of noisy answers on "Home & Kitchen" (top), and "Clothing, Shoes & Jewelry" (bottom) when $h(e)$ is modelled by term frequency. Our method, when the optimal β is being used, outperforms the best query-product matching baselines, presented in Table 3.6, after about 4 questions are being asked, despite the noise in the answers.

	MRR			Recall @ 5			NDCG		
# of questions asked	No-noise	optimal β	β=0	No-noise	optimal β	β=0	No-noise	optimal β	β=0
5	0.292	0.156	0.129	0.439	0.232	0.185	0.423	0.281	0.248
10	0.684	0.245	0.194	0.809	0.325	0.264	0.749	0.358	0.308
15	0.860	0.295	0.220	0.923	0.376	0.283	0.890	0.401	0.329
20	0.932	0.330	0.250	0.965	0.400	0.308	0.947	0.431	0.356
	MRR			Recall @ 5			NDCG		
# of questions asked	No-noise	optimal β	β=0	No-noise	optimal β	β=0	No-noise	optimal β	β=0
5	0.145	0.061	0.048	0.213	0.090	0.069	0.288	0.176	0.160
10	0.486	0.095	0.068	0.645	0.133	0.094	0.588	0.209	0.181
15	0.719	0.112	0.091	0.835	0.152	0.123	0.777	0.224	0.204
20	0.819	0.126	0.106	0.906	0.170	0.142	0.859	0.239	0.217

noise in the answers.

3.4.6 **RQ2.5** The Effectiveness of QSBPS Compared with Other Algorithms

In **RQ2.5**, we answer how effective is QSBPS for finding the best matching product compared to the six baselines shown in Section 3.4.1, by reporting NDCG and Recall@5. Table 3.6 shows the results. For QSBPS, for each number of questions asked, the near-optimal value of γ was used as indicated by the white-boundary boxes and tables in Figure 3.2. As indicated in Table 3.6, our QSBPS algorithm achieves the highest effectiveness scores when compared to Random, LSE, SBS, TranSearch, ALSTP, and PMMN. Our QSBPS algorithm exceeds Random, which indicates, as expected, that our question selection strategy is better than choosing questions in random. After less than 5 questions, our QSBPS algorithm greatly improves over the LSE, TranSearch, and ALSTP models. This suggests that a theoretically optimal sequence of entity-centered questions can be rather helpful and greatly improve the retrieval performance in product search. Our QSBPS algorithm performs better than SBS, which indicates the effectiveness of our cross-user duet training. Finally, our QSBPS model outperforms the interactive PMMN system. This can be explained by the fact that our pool of questions is better or the fact that our question sequence strategy is better. Given that in both systems, QSBPS and PMMN, the question strategy is strongly connected to the type of questions placed in the question pool, it is hard to decompose the effect in the improvements demonstrated by QSBPS. One final observation on PMMN is that its performance almost does not increase when the number of questions is larger than 10.

Table 3.6: NDCG and Recall@5 on "Home & Kitchen" (top), and "Clothing, Shoes & Jewelry" (bottom) for the compared methods. #. represents the number of questions asked.

	NDCG							Recall @ 5						
#.	Random	LSE	TranSearch	ALSTP	PMMN	SBS	QSBPS	Random	LSE	TranSearch	ALSTP	PMMN	SBS	QSBPS
5	0.011	0.176	0.198	0.181	0.271	0.001	**0.401**	0.006	0.074	0.131	0.149	0.158	0	**0.439**
10	0.012	0.176	0.198	0.181	0,285	0.267	**0.742**	0.006	0.074	0.131	0.149	0.188	0.267	**0.809**
15	0.015	0.176	0.198	0.181	0.286	0.815	**0.888**	0.010	0.074	0.131	0.149	0.188	0.828	**0.923**
20	0.014	0.176	0.198	0.181	0.286	0.923	**0.946**	0.009	0.074	0.131	0.149	0.188	0.919	**0.965**
25	0.015	0.176	0.198	0.181	0.286	0.960	**0.966**	0.008	0.074	0.131	0.149	0.188	0.961	**0.977**
30	0.016	0.176	0.198	0.181	0.286	0.980	**0.985**	0.011	0.074	0.131	0.149	0.188	0.981	**0.984**
	NDCG							Recall @ 5						
#.	Random	LSE	TranSearch	ALSTP	PMMN	SBS	QSBPS	Random	LSE	TranSearch	ALSTP	PMMN	SBS	QSBPS
5	0	0.098	0.101	0.131	0.231	0.001	**0.256**	0	0.045	0.056	0.083	0.101	0	**0.213**
10	0	0.098	0.101	0.131	0.248	0.178	**0.577**	0	0.045	0.056	0.083	0.13	0.075	**0.645**
15	0.001	0.098	0.101	0.131	0.249	0.632	**0.772**	0.001	0.045	0.056	0.083	0.13	0.664	**0.835**
20	0.001	0.098	0.101	0.131	0.249	0.817	**0.856**	0.001	0.045	0.056	0.083	0.13	0.833	**0.906**
25	0.001	0.098	0.101	0.131	0.249	0.895	**0.915**	0.002	0.045	0.056	0.083	0.13	0.904	**0.928**
30	0.001	0.098	0.101	0.131	0.249	0.931	**0.943**	0.001	0.045	0.056	0.083	0.13	0.944	**0.960**

The reason for this is that it is rather difficult to extract more than 10 aspect-value pairs from each user review for a certain item. As a consequence, there are no more available questions to ask.

3.5 Conclusions and Discussion

The focus of this chapter is on helping users to find the most relevant product in a large repository. We propose a question-based sequential Bayesian product search algorithm, called QSBPS, which efficiently locates the most relevant product by directly querying users on the presence or absence of an informative term in product related documents. Our framework first identifies and extracts informative terms (instantiated by entities in this chapter) mentioned in the text of the given product, and then trains a system belief and question reward model by using historical data. Based on the trained model, our method derives a policy for the optimal questions to be asked to the user. After receiving an answer to each question asked to the user, the posterior product preference of the user is calculated to generate the ranked recommendation list. Experiments on the Amazon product dataset show the effectiveness of QSBPS compared to state-of-the-art. Further, we illustrate the performance of our method when noisy answers are received by users.

In this chapter we pivot around the presence or absence of entities in the target products to generate a pool of questions to be asked to the user, which is still a rudimentary method of generating questions. Clearly, there is a richer set of possible questions to be asked, questions that may or may not be answered by a "yes" or a "no". Questions similar to the ones we have constructed could also be constructed by using labelled topics [223], keyword extraction [23], item categories and attributes, or extracted triplets [148]. A richer type of questions could also be constructed by identifying properties of the products in the descriptions and reviews and their relation to the product. For example, for a "Canon EOS 5D Mark II" digital camera, the following relations could be identified in the product description: "resolution", "manufacturer", "LCD screen dimension". We leave this as future work. Further, our work simulates user answers, noisy or not. A user study can be particularly helpful in understanding whether users are willing to answer a small number of questions, under what conditions (e.g., they may be willing if they have already reformulated their query a number of times), and to what extent they can provide correct answers. From a technical perspective, our work proposes a stand-alone algorithm that learns the informativeness of questions, along with user preferences. In principle, however, one can use a ranking method (any of the baselines) to construct an informative prior belief on user preferences and reduce the number of necessary questions to find the product to smaller than 5. Further, one can also incorporate other factors (e.g, the importance level of different informative terms) to the objective function of question selection to extend the work. Furthermore, in this chapter we made the assumption that we know the topic of a user's query, so that we can load the right prior over preferences, and entity rewards. In practice, one needs some technique (of text similarity) to soft-match an arbitrary query to the already known queries, which we intent to explore in the future. Last, it is highly likely that an entity is semantically related to the desired product but it is not lexically contained in the

description of it. In this chapter we do not explore any semantic correlation modeling, but we leave it as the future work.

In this chapter, to answer **RQ2**, we have proposed a question-based approach to assist product search and validate the effectiveness of asking CQs in product search by the proposed question-based approach. Also, on top of Chapter 2, we demonstrate that our duet learning framework involving question reward learning is highly beneficial. Next, we explore the application by asking CQs for recommendation.

4 Question-based Recommendation

In this chapter, we propose a novel question-based recommendation algorithm based on traditional matrix factorization, to assist users to find items interactively, by asking automatically constructed and algorithmically chosen clarifying questions (CQs). Like Chapter 2 and Chapter 3, we ask CQs about entities and utilize Generalized Binary Search (GBS) to select CQs to ask to the user. Different from Chapter 2 and Chapter 3 that learn a belief over document relevance or product relevance to retrieve documents or products, in this chapter, we incorporate user feedback into a novel matrix factorization model and use the output of matrix factorization to recommend items. We answer the following research question:

RQ3 How can we effectively ask CQs to improve recommender system performance?

4.1 Introduction

Online shopping platforms, such as Amazon, and eBay, is increasingly prevalent, and helps customers make purchase decisions [219]. The high demand for online shopping calls for task-oriented conversational agents that can interact with customers, helping them find items or services effectively [170]. This stimulates related research on conversational and question-based recommender systems [170, 213].

Traditional recommender systems infer user preferences based on their historical behavior, with the assumption that users have *static* preferences. Unfortunately, user preferences might evolve over time due to internal or external factors [142]. Besides, the quality of traditional recommendations suffers greatly due to the *sparsity* of users' historical behavior [163]. Even worse, traditional recommendation systems such as collaborative filtering fail to generate recommendations for new users or new items, for which the historical data is entirely missing: the *cold-start* problem [163]. Compared to the traditional approaches, question-based and conversational recommender systems overcome these issues by placing the *user in the recommendation loop* [170, 213]. By iteratively asking questions and collecting feedback, more accurate recommendations can be generated for the user.

Work on conversational and question-based recommenders [31, 113, 170, 213] demonstrates the importance of interactivity. Christakopoulou et al. [31] presented a

This chapter was published as [225].

recommender system that elicits user preferences over items. Sun and Zhang [170] and Li et al. [113] train their models on a large number of natural language conversations, either on the basis of predefined and well-structured facets [170] or based on free-style dialogues but require dialogues to mention items [113]. Zhang et al. [213] proposed a unified paradigm for product search and recommendation, which constructs questions on extracted item aspects, and utilizes user reviews to extract values as simulated user answers. While these publications have developed a successful direction towards conversational recommendation, research in the field is still limited. Christakopoulou et al. [31] collect user preferences over items, which is inefficient when the item pool is large and continuously updated. Sun and Zhang [170], Zhang et al. [213] and Li et al. [113] make certain assumptions over their input data, most importantly the availability of historical conversational data, or the availability of hierarchical item facets and facet-value pairs. In our work, we drop these assumptions: we only hypothesize that items can be discriminated based on textual information associated with them, e.g., descriptions and reviews [219, 224]. Our model asks questions based on extracted descriptive terms in the related contents, and beliefs are updated based on collaborative filtering, which is one of the most successful technologies in recommender systems [83, 163].

In this chapter, we propose a novel **Q**uestion-based **rec**ommendation method, Qrec, to assist users to find items interactively.[1] Our proposed model

(1) follows the work by Zou et al. [224] and Zou and Kanoulas [219], and generates questions over extracted informative terms; a question pool is constructed by entities (informative terms) extracted from the item descriptions and reviews.

(2) proposes a novel matrix factorization method to initialize the user and item latent factors offline by using user-item ratings;

(3) develops a belief-updating method to track the user's belief (preferences over items), and uses GBS [131] to select a sequence of questions based on the tracked user belief, aiming at learning to ask discriminative questions to gain new information about the user;

(4) asks questions, receives answers, updates the user and item latent factors online accordingly by incorporating feedback from the user based on our proposed matrix factorization algorithm, and also renews the user belief to select the next question to ask;

(5) generates a recommendation list based on the final user and item latent factors.

Our model combines the advantages of collaborative filtering based on matrix factorization and content analysis by querying users about extracted informative terms. The matrix factorization model is able to utilize the rating data and discover *latent* correlations between items, while incorporating question-answering over content information provides *explicit* content discrimination to assist the recommender systems. By iteratively asking questions over informative terms and collecting immediate feedback from the user, our question-based recommender can *track shifted user preferences*, clarify user needs, and improve capturing the true underlying user latent factors and

[1] Source code: https://github.com/JieZouIR/Qrec

item latent factors. Besides, the information gathered from the user constitutes new observations to *overcome the sparsity and cold-start problem*.

The main contribution of this chapter is three-fold:

- We propose a novel question-based recommendation method, Qrec, that interacts with users by soliciting their preferences on descriptive item characteristics.
- We propose a novel framework, that incorporates online matrix factorization and online users' belief tracking for sequential question asking.
- We propose a novel matrix factorization method that can incorporate offline training and efficient online updating of user and item latent factors.

To the best of our knowledge, this is the first work that incorporates online matrix factorization and question asking for item related features. The evaluation results show that the Qrec model achieves the highest performance compared to state-of-the-art baselines and that the model is effective in both user and item cold-start recommendation scenarios.

4.2 Related Work

Recommender systems can be classified into three categories: content-based [138], collaborative filtering [83, 94], and hybrid [214] systems. Conversational and question-based recommender systems can extend recommender systems in any of the three categories. Early related attempts include the work by Bridge [21], Carenini et al. [24], Felfernig et al. [59], Mahmood and Ricci [124, 125], Thompson et al. [175]. More recently, different ways of feedback are introduced [31, 67, 91, 121, 160, 195, 202, 215]. Zhao et al. [215] studied the problem of interactive collaborative filtering, and proposed methods to extend Probabilistic Matrix Factorization (PMF) [130] using linear bandits to select the item to ask feedback for and incorporate the rating back to the PMF output. Loepp et al. [121] focused on set-based feedback, while Graus and Willemsen [67] focused on choice-based feedback to learn latent factors and perform interactive preference elicitation online. Contrary to these publications that update the individual user's latent representation, Christakopoulou et al. [31] proposed a method to update all user and item latent factor parameters of a PMF variant at every feedback cycle, obtaining absolute and pairwise user feedback on items. We refer the reader to He et al. [82] and Jugovac and Jannach [92] for a literature review of interactive recommendation. Compared with Christakopoulou et al. [31], our model also updates all user and item latent factor parameters but based on our own matrix factorization model. Further, while Christakopoulou et al. [31] elicit user ratings on items, the Qrec model asks questions about extracted descriptive terms of the items, and learns a strategy of asking sequential questions. Furthermore, the selection of questions in Qrec is adaptive to the change of user preferences, instead of relying on the distribution of the items [31]. Last, Christakopoulou et al. [31] focus on rating prediction while our work focus on the top-N recommendation. They use semi-synthetic data for which they need to obtain the ground truth of the user's preference to every item (like/dislike) using bootstrapping, and thus simulate user's answers for each question, which is not available in our case.

Zhang et al. [213] designed a unified framework for product search and recommendation, and proposed a Personalized Multi-Memory Network (PMMN) architecture for conversational search and recommendation by asking questions over "aspects" extracted from user reviews by the means of sentiment labeling. Their model obtains the opinion of the user (i.e., value of the aspect-value pair) for the "aspect" as feedback. They utilize the user query as an initial query and use the aspect-value pairs of the conversation to expand the representation of the user's query, and thus to match the search and recommendation results. Different from this work that only uses the content of user reviews, we incorporate user ratings by collaborative filtering based on our proposed matrix factorization model. Besides, their work trains a model using the data for each user while our online question answering can work without this training data for cold-start users and items. Moreover, they query the aspect-value pairs extracted from user review and choose questions based on the log-likelihood of probability estimation over aspects, while we ask questions about descriptive terms of items and select questions based on the user belief tracking and GBS.

Reinforcement learning and deep learning on dialogue agents have also been studied for recommendations [27, 32, 68, 111, 114]. Sun and Zhang [170] proposed a deep reinforcement learning framework to build a conversational recommendation agent, which queries users on item facets and focuses on the long-term utility of success or conversion rate. Li et al. [113] presented a publicly available dataset called ReDial, and explored a neural method for composing conversational recommendations. They predict user opinions over the mentioned items based on the dialogue and sentiment classification to generate a recommendation. On the basis of the ReDial dataset, Chen et al. [27] proposed a knowledge-based recommender dialog system framework, which incorporates a recommender into a dialog system by using knowledge graphs and transformers. The aforementioned work train on usage data (i.e., existing natural language conversations or interactions with the recommender system). Sun and Zhang [170] require a large number of repeated interactions between the users and the information seeking system to train upon, while Li et al. [113] and Chen et al. [27] require mentioning items during the natural language dialogue. Such data is not always available. Different from these publications, our method does not require such data with large numbers of repeated interactions and mentioned items.

Learning to ask is another recent and related field of study [87, 190]. Hu et al. [87] presented a policy-based reinforcement learning method to identify the optimal strategy of question selection by continuously learning the probability distribution over all the objects on a 20 Questions game setup. They regard the learned probability distribution on confidence as a state and select the next question according to this state. Different from our work, their work introduces data-hungry techniques, which require having large numbers of labeled data and repeated interactions from multiple users for a target item to train upon. A recent line of work that also involves learning to ask is the work in dialogue and information seeking conversational systems [5, 28]. For example, Wang et al. [190] studied how to ask good questions in large-scale, open-domain conversational systems with neural question generation mechanisms. These models need to be trained on existing natural language conversations, which is different from our setup that depends on user ratings. Zou and Kanoulas [219] proposed an interactive sequential Bayesian model for product search. They learn to ask a good

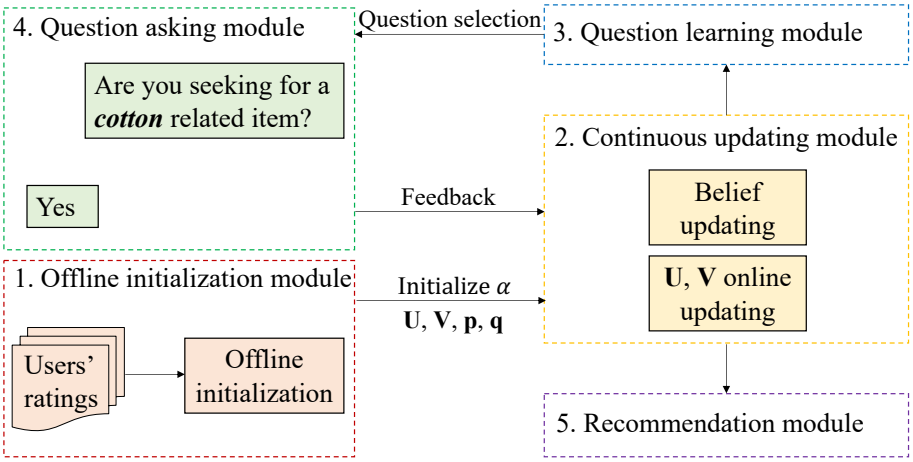


Figure 4.1: Framework of our proposed question-based recommendation model, Qrec. *Cotton* is an extracted entity (informative term), $\mathbf{U}$, $\mathbf{V}$, $\mathbf{p}$, $\mathbf{q}$ are model variables, and α is a hyper-parameter of user belief.

question by cross-user duet training, which learns a belief over product relevance and the rewards over question performances. Different from their work, which focuses on a sequential Bayesian product search model based on a cross-user duet training, our model incorporates user feedback into a matrix factorization model for recommendation. Further, they require the question answering history and purchase behavior from the same input query for their duet training, while our model does not require having such data.

4.3 Methodology

In this section, we discuss how we build our question-based recommender system. Our framework shown in Figure 4.1 comprises of five modules: (1) an offline initialization module (Section 4.3.1); (2) a continuous updating module (Section 4.3.1); (3) a question learning module (Section 4.3.2); (4) a question asking module (Section 4.3.3); and (5) a recommendation module (Section 4.3.4).

4.3.1 Latent Factor Recommendation

In this section, we describe two of the subcomponents of the Qrec model (shown in Figure 4.1): the offline initialization module and the continuous updating module.

Let $\mathbf{R} \in \mathbb{R}^{N \times M}$ be a user-item matrix, and let $\mathbf{R}_{i\cdot}$ represent the i-th row of $\mathbf{R}$, $\mathbf{R}_{\cdot j}$ represents the j-th column of $\mathbf{R}$. Here, N and M are the number of users and the number of items, respectively. Similarly, we use $\mathbf{Y}_{i\cdot}$ to represent the i-th row of our online affinity matrix $\mathbf{Y} \in \mathbb{R}^{N \times M}$, which is for incorporating user feedback (which will be discussed later), and use $\mathbf{Y}_{\cdot j}$ to represent the j-th column of $\mathbf{Y}$. $\mathbf{U} = [\mathbf{u_1}, \mathbf{u_2}, \ldots, \mathbf{u_i}, \ldots, \mathbf{u_N}]$, $\mathbf{V} = [\mathbf{v_1}, \mathbf{v_2}, \ldots, \mathbf{v_j}, \ldots, \mathbf{v_M}]$, where $\mathbf{u_i}$, $\mathbf{v_j}$ are user and

item latent factors, respectively. $\mathbf{u_i}$ and $\mathbf{v_j}$ are column vectors. Unless mentioned otherwise, all the vectors in this chapter are column vectors. $\mathcal{D}$ is the item collection represented by item documents (descriptions and reviews).

Matrix factorization recommendation techniques have proven to be powerful tools to perform collaborative filtering in recommender systems [103]. Assume we have N users and M items, matrix factorization decomposes a partially-observed matrix $\mathbf{R} \in \mathbb{R}^{N \times M}$ into two low-rank matrices, the user latent factors $\mathbf{U} \in \mathbb{R}^{N \times K}$ and the item factors $\mathbf{V} \in \mathbb{R}^{M \times K}$ where K is the dimension of user and item latent factors. The prediction of the unobserved entries in $\mathbf{R}$ is performed as a matrix completion, i.e., $\mathbf{R} \approx \mathbf{U}\mathbf{V}^\top$. Matrix factorization-based methods have been proposed and successfully applied to various recommendation tasks [31, 94, 103, 130]. In matrix factorization, users and items are mapped to the same latent space. Items that have been co-liked by users will lie close in a low dimensional embedding space (latent vector).

In this chapter, we propose a novel model to perform the matrix factorization recommendation, and we refer to it as QMF. The generative process for our model is:

1. For each user $i = 1, \ldots, M$, draw a user latent factor $\mathbf{u_i} \sim \mathcal{N}(0, \lambda_u^{-1}\mathbf{I})$;
2. For each item $j = 1, \ldots, N$, draw an item latent factor $\mathbf{v_j} \sim \mathcal{N}(0, \lambda_v^{-1}\mathbf{I})$.
3. For each user-item pair $(i, j) \in \mathbf{R}$, draw $R_{ij} \sim \mathcal{N}(\mathbf{p}^\intercal(\mathbf{u_i} \circ \mathbf{v_j}), 1)$.
4. In each user session targeting a certain item, for each user-item pair $(i, j') \in \mathbf{Y}$, draw $Y_{ij'} \sim \mathcal{N}(\mathbf{q}^\intercal(\mathbf{u_i} \circ \mathbf{v_{j'}}), \gamma^{-1}\mathbf{I})$ for each question asked.

In the above, λ_u, λ_v are hyper-parameters modeling the variances in latent vectors, and γ is a hyper-parameters modeling the variance in $Y_{ij'}$. $\mathbf{p}$ and $\mathbf{q}$ are the free parameters of column vector of dimension K for R_{ij} and Y_{ij}, respectively. The intuition behind this is that $\mathbf{p}$ and $\mathbf{q}$ can capture some general information across users and items.

Optimization When optimizing QMF, the maximization of the posterior distributions over $\mathbf{U}$ and $\mathbf{V}$ can be formulated as follows according to the generative process:

$$\max_{\mathbf{U},\mathbf{V},\mathbf{p},\mathbf{q}} p(\mathbf{U}, \mathbf{V}, \mathbf{p}, \mathbf{q} | \mathbf{R}, \mathbf{Y}, \lambda_u, \lambda_v, \lambda_p, \lambda_q, \gamma). \tag{4.1}$$

Then the maximization of the posterior probability can be reformulated as the minimization of its negative logarithm, which is

$$\begin{aligned} & -\log p(\mathbf{U}, \mathbf{V} \mid \mathbf{R}, \mathbf{Y}, \Theta) \\ \propto \; & \frac{1}{2} \sum_{i,j \in \mathbf{R}} (R_{ij} - \mathbf{p}^\intercal(\mathbf{u_i} \circ \mathbf{v_j}))^2 + \frac{\gamma}{2} \sum_{i,j \in \mathbf{Y}} (Y_{ij} - \mathbf{q}^\intercal(\mathbf{u_i} \circ \mathbf{v_j}))^2 + \\ & \sum_{i=1}^{M} \frac{\lambda_u}{2} |\mathbf{u_i}|_2^2 + \sum_{j=1}^{N} \frac{\lambda_v}{2} |\mathbf{v_j}|_2^2 + \frac{\lambda_p}{2} |\mathbf{p}|_2^2 + \frac{\lambda_q}{2} |\mathbf{q}|_2^2, \end{aligned} \tag{4.2}$$

where $\Theta = \{\mathbf{p}, \mathbf{q}\}$ are the parameters, and γ is a trade-off of the online affinity $\mathbf{Y}$ for incorporating the user feedback.

Offline Optimization When optimizing offline by using historical ratings of all users, we use gradient descent with the Adaptive Moment Estimation (Adam) optimizer [100] for Equation 4.2, with γ set to 0, since we do not have historical question asking data and thus do not have Y_{ij} for the question asking. Therefore, we do not train $\mathbf{q}$, instead set $\mathbf{q}$ to an all-ones vector in this chapter, but one can also train $\mathbf{q}$ using historical question asking data. That is, the model variables $\mathbf{U}, \mathbf{V}, \mathbf{p}$ are learned by maximizing the log-posterior over the user and item latent vectors with fixed hyper-parameters, given the training observations $\mathbf{R}$.

Online Optimization Since we aim to recommend items online, it is necessary to update the variables effectively and efficiently according to the user feedback. Thus, we optimize Equation 4.2 by Alternating Least Square (ALS) technique to update the model variables $\mathbf{u_i}$, and $\mathbf{v_j}$ in order to guarantee efficiency. Then we have our following derived closed-form solution:

$$\mathbf{u_i} = \left(\mathbf{V_p}^\intercal \mathbf{V_p} + \gamma \mathbf{V_q}^\intercal \mathbf{V_q} + \lambda_u \mathbf{I}\right)^{-1} \left(\mathbf{V_p}^\intercal \mathbf{R}_i + \gamma \mathbf{V_q}^\intercal \mathbf{Y}_i\right) \tag{4.3}$$

$$\mathbf{v_j} = \left(\mathbf{U_p}^\intercal \mathbf{U_p} + \gamma \mathbf{U_q}^\intercal \mathbf{U_q} + \lambda_v \mathbf{I}\right)^{-1} \left(\mathbf{U_p}^\intercal \mathbf{R}_{\cdot j} + \gamma \mathbf{U_q}^\intercal \mathbf{Y}_{\cdot j}\right) \tag{4.4}$$

where

$$\begin{aligned}
\mathbf{V_p} &= \mathbf{V}\text{diag}(\mathbf{p}), \\
\mathbf{V_q} &= \mathbf{V}\text{diag}(\mathbf{q}), \\
\mathbf{U_p} &= \mathbf{U}\text{diag}(\mathbf{p}), \\
\mathbf{U_q} &= \mathbf{U}\text{diag}(\mathbf{q}).
\end{aligned}$$

ALS repeatedly optimizes one of $\mathbf{U}$ and $\mathbf{V}$ while temporarily fixing the other to be constant. After each question being asked and feedback received, we update $\mathbf{U}$ and $\mathbf{V}$. We assume that there is a target item related document $d^* \in \mathcal{D}$ and define an indicator vector y_j^l for the l-th question, with each dimension j corresponding to an item in the collection:

$$y_j^l = \mathbb{1}\{e_l^{d_j} = e_l^{d^*}\}, \tag{4.5}$$

$$Y_j = \sum_{t=0}^{l-1} y_j^t, \tag{4.6}$$

where $e_l^{d_j}$ is true if the item related document d_j contains the l-th requested entity e_l (see Section 4.3.3 for details about the question construction), and $\mathbb{1}\{\cdot\}$ is an indicator function. $e_l^{d^*}$ expresses whether the target item contains the l-th requested entity e_l. This also represents the answer by the user, given that the user's answers are driven by a target item. Hence, for example if the question is "Are you seeking for a [cotton] item?" and the target item description includes "cotton" as an entity, then y_j^l is 1 for all items that also have "cotton" as an important entity. If the question is "Are you seeking for a [beach towel] item?" and the target product does not contain a "beach towel" in its description or reviews (hence, the answer of the user is "no"), then y_j^l is 1 for all the items that are not beach towels. Y_j is the accumulated y_j with the dimension corresponding to j-th item until the l-th question.

Based on whether or not the target item is relevant to the requested entity, the feedback from the user becomes a new or an updated observation for our system, and

hence it is used to update $\mathbf{Y}$ related to the particular user, i.e., $\mathbf{Y}_i$, which is a vector of the online affinity for user i, with each of the dimension Y_{ij} corresponding to j-th item. Then $\mathbf{u_i}$, and all item factors $\mathbf{V}$ are updated by Equation 4.3 and Equation 4.4. Note that this observation only affects the current user's interaction session, and not any follow-up user interactions. As we ask about an entity e and observe the user's response, the user's preference over the items which are consistent with the answer increases. The variance of the inferred noisy preferences over these items which are consistent with the answer, as well as the variance of preferences over the nearby items in the learned embedding, are reduced. The model's confidence in its belief over the user's preference on these items increases. As the system keeps asking questions to user i and incorporates his/her responses, the latent user feature vectors $\mathbf{U}$ and latent item feature vectors $\mathbf{V}$ change and move towards the true underlying user and item latent vectors.

After updating our matrix factorization model, we use the final user latent factor $\mathbf{U}$ and item latent factor $\mathbf{V}$ to computing $\mathbf{U}\mathbf{V}^\top$ to yield a ranking of items to generate the recommendation list, which constitutes the recommendation module in Figure 4.1.

4.3.2 Question Learning

In this section, we describe how we select the next question to ask from the question pool (see Section 4.3.3 for the question pool construction). After the offline initialization by using all of the historical ratings, the user initiates an interaction with our recommender system, our system asks a few questions to learn about the user latent factor, the item latent factor, and the user's belief. During this interactive phase, it is important to select the most informative questions that lead to learning effectively the user's preference, so as to minimize the number of questions asked and locate the target item effectively.

Similar to Wen et al. [193] and Zou et al. [224], we use the estimated user preferences to help the question learning module to learn the most discriminative question to ask next. We model the user preferences for the items by a (multinomial) probability distribution π^* over items $\mathcal{D}$, and the target item is drawn i.i.d. from this distribution. We also assume that there is a prior belief $\mathbb{P}$ over the user preferences π^*, which is a probability density function over all the possible realizations of π^*:

$$\mathbb{P}_l = Dir(\alpha + \mathbf{Y}_i), \tag{4.7}$$

where $\mathbb{P}$ is a Dirichlet distribution with parameter α. Having applied the offline initialization of our matrix factorization model, items can be scored and ranked for each user, the rank of each item expresses our initial belief on the preference of items for each given user. This initial belief will be used to initialize the hyper-parameter α of the Dirichlet distribution. In particular, we set α_i for item i to $1/(p_i + 1)$, where p_i is the index of item i in the ranked list. $\mathbf{Y}_i$ is the vector for the user i with each dimension corresponding to accumulated y_j^l until the l-th question.

Let $\mathbb{P}_l$ be the system's belief over π^* prior to the l-th question. We compute the user preferences $\pi_l^*(d)$ prior to the l-th question by:

$$\pi_l^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] \quad \forall d \in \mathcal{D}. \tag{4.8}$$

The π^* is a multinomial distribution over items $\mathcal{D}$, and $\mathbb{P}$ is modeled by the conjugate prior of the multinomial distribution, i.e., the Dirichlet distribution. From the properties

of the Dirichlet distribution, the user preferences π_l^* can be updated by counting and re-normalization of α and $\mathbf{Y}_i$. As the system keeps asking questions to the user and incorporates his/her response, the predicted belief and preferences about the user is updated accordingly. This belief tracker thus specifies the direction for moving towards the true underlying belief distribution and true user preferences. These predicted user preferences will be used for guiding the question selection.

Like Wen et al. [193] and Zou et al. [224], we apply GBS to find the entity that best splits the probability mass of predicted user preferences closest to two halves for the remaining of the items during the l-th question, as the nearly-optimal entity to ask:

$$e_l = \arg\min_e \Big| \sum_{d \in \mathcal{C}_l} (2\mathbb{1}\{e^d = 1\} - 1)\pi_l^*(d) \Big|, \tag{4.9}$$

where e_l is the l-th chosen entity, $\mathcal{C}_l$ is the candidate version space containing the set of remaining items when asking the l-th question; the initial $\mathcal{C}_l$ is equal to $\mathcal{D}$, e^d expresses whether the item d contains the entity e or not. Specifically, for the entity embedding in this chapter, the entity is represented by a one-hot encoding, i.e., if the entity appears in a certain item document, the value of the dimension corresponding to this item is 1 ($e^d = 1$), otherwise the value of the dimension corresponding to this item is 0 ($e^d = 0$). After each question is asked and the answer is obtained, the user preferences π^* are updated by the belief tracker module. GBS tends to select entities by minimizing the objective function of Equation 4.9. This means that GBS selects the entity that is able to split the sum of calculated user preferences corresponding to the item with $e^d = 1$ and the sum of user preferences corresponding to the item with $e^d = 0$ closest to two halves.

4.3.3 Question Asking

The proposed method of learning informative questions to ask to users, depends on the availability of a pool of questions regarding informative terms. Given an item, the user should be able to answer questions about this item with a "yes" or a "no", having a reference to the relevant item (or item in mind).

In this chapter, we use the approach taken by Zou et al. [224], and Zou and Kanoulas [219] to extract meaningful short-phrases – typically entities – from the surface text to construct the question pool using the entity linking algorithm TAGME [61]. These entities are recognized to comprise the most important characteristics of an item [219, 224], and we generate questions about the presence or absence of these entities in the item related documents. One could also use other sources like labelled topics, extracted keywords, item categories and attributes, to construct questions.

In TAGME, each annotated short-phrase in unstructured text is weighted using a probability, that measures the reliability of that substring being a significant mention. Only the short-phrases with high probability should be considered as entities. In this chapter, similar to Ferragina and Scaiella [61], and after a set of preliminary experiments, we set the threshold to 0.1 and filter out short-phrases whose probability is below 0.1. Prior to this, we also removed stop words such as "about", "as well" etc..

Having extracted the most important entities from the corpus, the proposed algorithm asks a sequence of questions of the form "Are you seeking for a [entity] related item?"

Algorithm 5: The proposed Qrec algorithm

input : A item document set, $\mathcal{D}$, the set of annotated entities in the documents, $\mathcal{E}$, the ratings $\mathbf{R}$, number of questions to be asked N_q

1 $l \leftarrow 0$
2 $\mathbf{Y}_i \leftarrow \mathbf{0}$
3 Offline intialization of our matrix factorization model: $\mathbf{U}, \mathbf{V} = QMF(\mathbf{R})$
4 $Ranking_l = Sort(\mathbf{U}\mathbf{V}^\top)$
5 $\alpha \leftarrow Ranking_l$
6 **while** $l < N_q$ *and* $|\mathcal{C}_l| > 1$ **do**
7 Compute the user belief with α: $\mathbb{P}_l = Dir(\alpha + \mathbf{Y}_i)$
8 Compute the user preferences with $\mathbb{P}_l(\pi)$: $\pi_l^*(d) = \mathbb{E}_{\pi \sim \mathbb{P}_l}[\pi(d)] \quad \forall d \in \mathcal{D}$
9 Find the optimal target entity by question learning:
10 $e_l = \arg\min_e \left| \sum_{d \in \mathcal{C}_l} (2\mathbb{1}\{e^d = 1\} - 1)\pi_l^*(d) \right|$
11 Ask the question about e_l, observe the reply $e_l^{d^*}$
12 Remove e_l from the question pool
13 $\mathcal{C}_{l+1} = \mathcal{C}_l \cap \{d \in \mathcal{D} : e_l^d = e_l^{d^*}\}$
14 Update $\mathbf{Y}_i$ by the reply $e_l^{d^*}$ according to Equation 4.5 and Equation 4.6
15 Update $\mathbf{U}, \mathbf{V}$ by ALS according to Equation 4.3 and Equation 4.4
16 $l \leftarrow l + 1$
17 **end**
18 Generate recommendation list by updated $\mathbf{U}, \mathbf{V}$: $result = Sort(\mathbf{U}_{N_q} {\mathbf{V}_{N_q}}^\top)$

to locate the target item. In this case, the users can respond with a "yes", a "no" or a "not sure" according to their belief.

4.3.4 Question-based Recommender System

The algorithm of our question based recommender system is provided in Algorithm 5. The Qrec model performs two rounds: the offline phase and the online phase. The offline phase includes *line 3–5*, and the online phase includes *line 6–17* in Algorithm 5. During the offline phase, we first initialize our model parameters offline by using the history rating data across all users. We make the assumption that we have access to historical user-item interaction data (e.g., rating or purchasing data), even though the Qrec can work without it as well. When a new user session starts, we use the initialized user's latent factors and items' latent factors to yield the preliminary ranking of candidate items. We then utilize this ranking score to initialize the Dirichlet prior parameter α. We calculate the user belief with this α and $\mathbf{Y}_i$ in online phase. After that, we compute the user preferences with prior belief equal to $\mathbb{P}_l$, and find the optimal entity e_l by GBS. We ask whether the entity e_l is present in the target item that the user wants to find, d^*, observe the reply $e_l^{d^*}$, remove e_l from the question pool, and update the candidate version space $\mathcal{C}_l$. Then we update $\mathbf{Y}_i$ by the user response, and update the user latent factors $\mathbf{U}$ and the item latent factors $\mathbf{V}$ using ALS based on the updated $\mathbf{Y}_i$. After the online question asking phase is over, the recommendation list is generated by sorting the inner product of the last updated user latent factors $\mathbf{U}_{N_q}$ and item latent factors $\mathbf{V}_{N_q}$.

Table 4.1: Statistics of the dataset. #entity is the number of unique entities.

Dataset	#users	#items	#ratings	density	#entity
Home and Kitchen	9,124	557	10,108	0.20%	9,296
Pet Supplies	2,248	2,475	15,488	0.28%	71,074

4.4 Experiments and Analysis

4.4.1 Experimental Setup

Dataset. In our experiments we use a collection of Amazon items[2] [128]. Each item contains rich metadata such as title, descriptions, categories, and reviews from users as well. Following Van Gysel et al. [180] and Zou and Kanoulas [219], we use two product domains from the Amazon product dataset, which are "Home and Kitchen", and "Pet Supplies", respectively. The documents associated with every item consist of the item description and the reviews provided by Amazon customers. On the two item domains, we use the same item list[3] as Van Gysel et al. [180], and filtered those items and users that appeared in less than five transactions to construct the user-item recommendation matrix like most collaborative filtering papers [83]. We randomly split the entire dataset of user-item interactions into a training, validation and testing set with 80%, 10% and 10% splits similar to other recommendation papers, e.g., Sun and Zhang [170]. Statistics on the dataset are shown in Table 4.1.

Parameter Setting To learn the matrix factorization embedding, we set the hyper-parameters to the combination that achieved the highest pairwise accuracy in the offline observations: the maximum training iterations of PMF and our matrix factorization model is set to 100, and $\lambda_u = \lambda_v = \lambda_p = \lambda_q = 0.1$. The parameters γ, the dimension of the latent factors K, and the number of questions asked N_q are decided in **RQ3.1**.

Evaluation Metrics We use average Recall at cut-off 5 (recall@5), Average Precision at 5 (AP@5), and Mean Reciprocal Rank (MRR) and Normalized Discounted Cumulative Gain (NDCG) as our evaluation metrics, which are commonly used metrics for capturing accuracy in recommendation [31, 213, 215]. NDCG is calculated using the top 100 items, similar to other papers [180]. The ground truth used to compute the aforementioned metrics is constructed by looking at the historical buying behavior of the user; an item is considered relevant if the user wrote a review and gave it a rating, similar to other work [180, 213].

Baselines We compare the Qrec with five baselines; the first two are static baselines, while the other three are interactive baselines. In particular the baselines are: (1) **PMF**, which is a typical, static recommendation approach; (2) **NeuMF** [83], which is a state

[2] http://jmcauley.ucsd.edu/data/amazon/

[3] Product list: https://github.com/cvangysel/SERT/blob/master/PRODUCT_SEARCH.md

of the art approach for collaborative filtering and widely used as a baseline in other work. (3) **QMF+Random**, which uses our proposed matrix factorization for offline initialization and then randomly chooses a question from the question pool to ask; (4) **SBS**, which is the sequential Bayesian search algorithm. We applied the SBS [193] to our recommendation task and uses the same question asking strategy with the Qrec model, but with a uniform prior; and (5) **PMMN** [213], the Personalized Multi-Memory Network model, which is a state-of-the-art conversational recommender system asking questions on aspect-value pairs. For the PMF, QMF+Random, and SBS baselines, we use the same parameter setting as for the Qrec model. For the NeuMF and PMMN models, we use the optimal parameters reported in the corresponding papers and tuned their hyper-parameters in the same way as they reported.

Simulating Users Our experiments depend on users responding to questions asked by our method. In this chapter we follow recent work [170, 213, 219, 220, 224] and simulate users. We also conduct a small user study described next. During the simulation, we follow the approach proposed by Zou et al. [224] and Zou and Kanoulas [219], i.e., we assume that the user will respond to the questions with full knowledge of whether an entity is present or not in the target item. Hence, we assume that the user will respond with "yes" if an entity is contained in the target item documents and "no" if an entity is absent. This simulation also follows the one used by Zhang et al. [213], which assumes that the user has perfect knowledge of the value of an aspect for the target product.

Online User Study To confirm some of the assumptions made in this chapter and test how well our recommender system works "in-situ" we also conduct a small online user study. The ideal users would be ones who have actually bought a number of items on an online shopping platform and now converse with our system embedded in the platform to find their next target item. In the absence of such a user base and commercial recommender system we use a crowdsourcing platform. First, we let the crowd worker select a product category she feels familiar with. Then, we randomly sample a product from our test data as a target product. To let the user familiarize herself with the target product we provide her with a product image, title, description, and the entities extracted from the product reviews. After the user indicates that she is familiar with the product and the conversation with the system can start, the information of the target item disappears from the screen and a question is selected by our algorithm to be asked to the user. The user needs to provide an answer to the question according to the information she read in the previous step, and then our system updates according to the user answer. With each question being answered, the user is shown a grid (4-by-4) with pictures of the sixteen top ranked items. The user can stop answering questions any time during her interaction with the system. When stoping the interaction with the system, users are asked a number of exit questions about their experiences with the system.

Research Questions. Through the experiments in this chapter we decompose **RQ3** into the five following subquestions and aim to answer them:

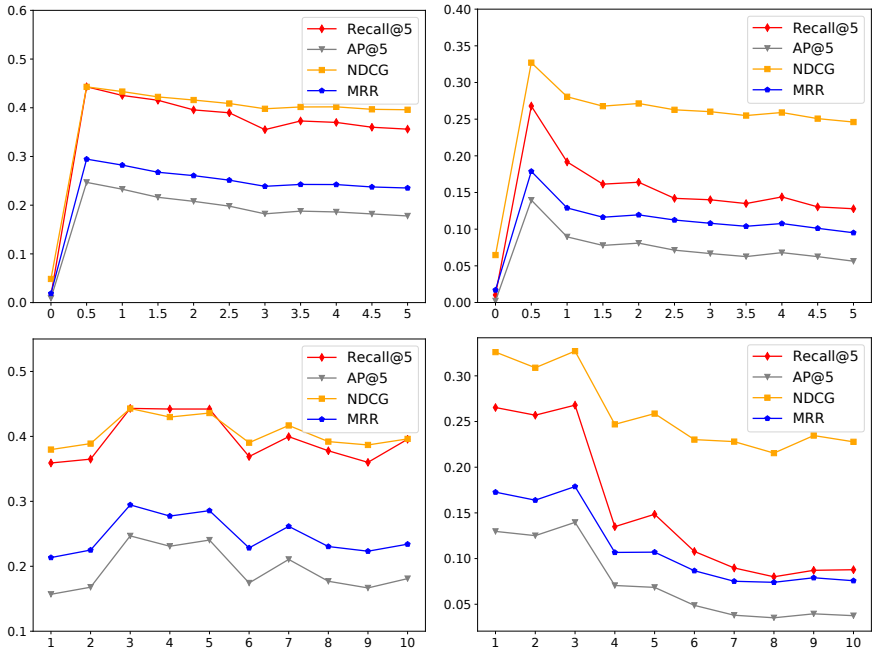


Figure 4.2: The impact of the trade-off parameter γ (top), and the dimension of the latent factors K (bottom) on "Home and Kitchen" (left) and "Pet Supplies" (right) categories.

RQ3.1 What is the impact of the trade-off γ, the dimension of the latent factors K, and the number of questions asked N_q?
RQ3.2 How effective is Qrec compared to prior works?
RQ3.3 How effective is Qrec for the cold-start user and the cold-start item problem?
RQ3.4 Does the offline initialization help?
RQ3.5 Are the assumptions made in this chapter along with the effectiveness of our algorithm confirmed by a user study?

4.4.2 **RQ3.1** Impact of Parameters

In **RQ3.1**, we examine the impact of the trade-off parameter γ, the dimension of the latent factors K, and the number of questions asked N_q over the effectiveness of our model. We compare the performance for different parameters. When focusing on one parameter, we fix the other two parameters. The performance of different values of γ and different dimensions of the latent factors K on the two categories is shown in Figure 4.2, and the results of different numbers of questions on the two categories can be seen in the "Qrec" column of Table 4.2. The values of γ ranges from 0 to 5 with a step of 0.5, and the K ranges from 1 to 10 with a step of 1. As one can observe, with the increase of γ, the recommendation performance improves first and then drops. The best performing γ is 0.5 on the two categories. γ can control how much online user feedback is incorporated into the user latent factor and item latent factor. In particular, when γ

Table 4.2: A comparison with PMF, NeuMF, QMF+Random, SBS, and PMMN on the "Home and Kitchen" (top) and the "Pet Supplies" (bottom) categories. #. represents the number of asked questions. QMF+Rand. represents the QMF+Randam model. Our proposed model achieve highest results when compared with interactive baselines, and our model performs better than the state of the art collaborative filtering model NeuMF on all of four different metrics with less than 5 questions.

	recall@ 5							AP@ 5						
#.	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec
2	0.011	0.062	**0.222**	0.075	0.060	0.073	0.130	0.004	0.037	**0.121**	0.047	0.028	0.021	0.072
5	0.011	0.062	0.222	0.095	0.353	0.194	**0.443**	0.004	0.037	0.121	0.064	0.170	0.091	**0.247**
10	0.011	0.062	0.222	0.121	0.883	0.216	**0.943**	0.004	0.037	0.121	0.088	0.661	0.105	**0.884**
15	0.011	0.062	0.222	0.151	0.933	0.216	**0.982**	0.004	0.037	0.121	0.117	0.863	0.105	**0.973**
20	0.011	0.062	0.222	0.188	0.962	0.216	**0.995**	0.004	0.037	0.121	0.144	0.911	0.105	**0.990**
	NDCG							MRR						
#.	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec
2	0.036	0.082	0.183	0.113	0.181	0.212	**0.215**	0.011	0.048	**0.136**	0.062	0.053	0.050	0.100
5	0.036	0.082	0.183	0.131	0.389	0.300	**0.443**	0.011	0.048	0.136	0.079	0.226	0.135	**0.295**
10	0.036	0.082	0.183	0.158	0.749	0.310	**0.915**	0.011	0.048	0.136	0.104	0.671	0.147	**0.889**
15	0.036	0.082	0.183	0.184	0.899	0.310	**0.980**	0.011	0.048	0.136	0.132	0.869	0.147	**0.975**
20	0.036	0.082	0.183	0.211	0.935	0.310	**0.993**	0.011	0.048	0.136	0.159	0.915	0.147	**0.991**
	recall@ 5							AP@ 5						
#.	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec
2	0.008	0.016	**0.214**	0.016	0.007	0.056	0.076	0.005	0.008	**0.119**	0.008	0.003	0.026	0.030
5	0.008	0.016	0.214	0.017	0.052	0.139	**0.268**	0.005	0.008	0.119	0.009	0.024	0.095	**0.140**
10	0.008	0.016	0.214	0.033	0.668	0.143	**0.966**	0.005	0.008	0.119	0.024	0.393	0.097	**0.770**
15	0.008	0.016	0.214	0.035	0.952	0.143	**0.999**	0.005	0.008	0.119	0.025	0.823	0.098	**0.997**
20	0.008	0.016	0.214	0.039	0.994	0.143	**1.000**	0.005	0.008	0.119	0.029	0.961	0.098	**1.000**
	NDCG							MRR						
#.	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec	PMF	QMF	NeuMF	QMF+Rand.	SBS	PMMN	Qrec
2	0.012	0.056	**0.179**	0.069	0.032	0.121	0.141	0.007	0.019	**0.134**	0.021	0.010	0.046	0.054
5	0.012	0.056	0.179	0.068	0.190	0.231	**0.327**	0.007	0.019	0.134	0.022	0.054	0.115	**0.179**
10	0.012	0.056	0.179	0.038	0.557	0.233	**0.830**	0.007	0.019	0.134	0.028	0.427	0.117	**0.774**
15	0.012	0.056	0.179	0.039	0.870	0.233	**0.998**	0.007	0.019	0.134	0.028	0.829	0.117	**0.997**
20	0.012	0.056	0.179	0.049	0.971	0.233	**1.000**	0.007	0.019	0.134	0.034	0.961	0.117	**1.000**

is 0, i.e., the online update does not take the user feedback (i.e., $\mathbf{Y}$) into account, as expected the performance is very bad. As for the dimension of the latent factors, K, the overall performance trend also rises and then goes down as K increases. This suggests that the dimension of the latent factors K should not be set too high or too low. In this chapter, we set it to the optimal value, i.e., 3. Unless mentioned otherwise, for remaining research questions, we use the optimal parameter γ, which is 0.5, and for K we used the optimal value 3. To figure out the impact of the number of asked questions, we vary N_q and inspect the performance shown in "Qrec" column of Table 4.2. As shown in Table 4.2, the performance of the Qrec model increases on all metrics with the increase of the number of questions, as expected. The more questions asked, the better the user needs are captured, and the closer the modeled user latent factor and item latent factor are to the true real-time user and item latent factors. Furthermore, the performance of Qrec reaches very good performance already, within the first 10 questions, while asking more than 10 questions does not add much regarding the performance.

4.4.3 **RQ3.2** Performance Comparison

To answer how effective is our proposed method compared to prior works, we compare our results with five baselines, PMF, NeuMF, QMF+Random, SBS, and PMMN. The results on the two categories are shown in Table 4.2. From Table 4.2, we can see that our proposed model, Qrec, achieves the highest results on all four metrics compared with the interactive baselines QMF+Random, SBS, and PMMN, on these two categories, which suggests that our question-based recommender system Qrec is effective. The Qrec model performs better than QMF+Random, this suggests that our question selection is effective. There are few fluctuations on some metrics for QMF+Random with different numbers of questions asked; this is because of the uncertainty of random question selection in different number of questions asked. The Qrec model outperforms the SBS model; this suggests that using the prior from the offline initialization is beneficial. We will further discuss this in **RQ3.4**. Further, the Qrec model performs better than PMMN [213], especially after 5 questions asked. This might be explained by the fact that asking questions on extracted entities can gather more information from users and is able to better learn user true preferences than asking questions on aspect-value pairs. Further, what we indeed observed is that the results of all four metrics regarding PMMN do not increase much and the differences in results between PMMN and Qrec become big when the number of questions is larger than 10. The reason for this is the fact that it is rather difficult to extract more than 10 aspect-value pairs from each user review for a certain item. As a consequence, there are no more available questions to ask, and thus the metric results never increase. Overall, this suggests that asking questions on extracted entities is more effective.

It can also be observed that our proposed matrix factorization model achieves better performance than PMF on the four metrics; this suggests that our proposed matrix factorization model is rather helpful. The reason might be that adding the parameter P improves the model capability of fitting. The NeuMF model outperforms the linear models PMF and QMF, this is because the nonlinear deep neural model can obtain more subtle and better latent representations. But note that the stacked neural network structures also make them difficult to train and incur a high computational

Table 4.3: Recommendation results on cold-start tuples. The top table represents the cold-start user tuples on "Home and Kitchen" and bottom table represents the cold-start item tuples on "Pet Supplies" category. The Qrec model can still achieve high performance for cold-start users and cold-start items.

# of questions	recall@5	AP@5	NDCG	MRR
PMF	0.005	0.002	0.039	0.011
2	0.127	0.071	0.215	0.099
5	0.448	0.245	0.442	0.293
10	0.944	0.883	0.914	0.889
15	0.985	0.974	0.981	0.976
20	0.996	0.991	0.994	0.992

# of questions	recall@5	AP@5	NDCG	MRR
PMF	0.000	0.000	0.000	0.000
2	0.000	0.000	0.008	0.003
5	0.046	0.011	0.157	0.035
10	0.853	0.561	0.676	0.576
15	0.991	0.961	0.971	0.962
20	1.000	1.000	1.000	1.000

cost. Specifically, our model is able to achieve better results than the NeuMF model on all of four metrics with less than 5 questions. With more questions being asked, the differences in results between NeuMF and Qrec become bigger. This shows that interactive or question-based recommendation can improve the performance over static models as interactive or question-based recommendation can continuously learn from the user.

4.4.4 **RQ3.3** Cold-start Performance Analysis

To explore if the Qrec is effective for the cold-start user and the cold-start item problem, we extract cold-start user tuples (i.e., user-item interactions in which the user has never appeared in the training set) and cold-start item tuples (i.e., user-item interactions in which the item has never appeared in the training set) from our test dataset. Because there are very few cold-start item tuples in "Home and Kitchen" category, and very few cold-start user tuples in the "Pet Supplies" category, to the extent that results would not be reliable, we only use cold-start user tuples from the "Home and Kitchen" category and cold-start item tuples from the "Pet Supplies" category to validate the cold-start addressing ability of our model. Statistics on the two categories shows that there are about 84% cold-start user tuples in the "Home and Kitchen" category and about 7% cold-start item tuples in the "Pet Supplies" category. The results on the two categories are shown in Table 4.3. As it is observed, the Qrec model can still achieve high recall@5, AP@5, NDCG, and MRR for cold-start users and cold-start items. As it is known, PMF does not really work for cold-start users and cold-start items, which is indeed what we observe. We conclude that the Qrec model is capable of tackling the cold-start recommendation problem.

Table 4.4: Results for the effects of offline initialization on the “Home and Kitchen” (top) and the “Pet Supplies” (bottom) categories. Qrec_offl. represents the Qrec including offline initialization, Qrec_rand. represents the Qrec with random initialization (i.e, excluding offline initialization). The Qrec including offline initialization is superior to the Qrec excluding offline initialization.

	recall@5		AP@5		NDCG		MRR	
# of questions	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.
2	0.13	0.08	0.07	0.04	0.22	0.19	0.10	0.07
5	0.44	0.34	0.25	0.17	0.44	0.39	0.29	0.22
10	0.94	0.93	0.88	0.88	0.91	0.91	0.89	0.89
15	0.98	0.98	0.97	0.97	0.98	0.98	0.97	0.97
20	1.00	0.99	0.99	0.99	0.99	0.99	0.99	0.99

	recall@5		AP@5		NDCG		MRR	
# of questions	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.	Qrec_offl.	Qrec_rand.
2	0.08	0.02	0.03	0.01	0.14	0.05	0.05	0.02
5	0.27	0.11	0.14	0.06	0.33	0.24	0.18	0.09
10	0.97	0.97	0.77	0.65	0.83	0.74	0.77	0.65
15	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
20	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Table 4.5: System effectiveness as determined in a user study. Results are in agreement with the Qrec results of Table 4.2.

# of questions	recall@5	AP@5	NDCG	MRR
5	0.333	0.082	0.305	0.129
10	0.848	0.717	0.777	0.727
15	0.879	0.760	0.806	0.762
20	0.909	0.775	0.820	0.776
stopping	0.939	0.790	0.834	0.791

4.4.5 **RQ3.4** Contribution of Offline Initialization

With this research question, we investigate the effect of our offline initialization. We compare the performance results including the offline initialization and the performance results excluding the offline initialization of our model (i.e., random initialization for the model parameters when the new user session starts). Our hypothesis is that the offline learned parameters from the historical ratings capture some general trend and provide a generic prior to guide the model. Indeed, the results shown in Table 4.4 demonstrates the model with offline initialization achieves higher performance than the one without offline initialization, especially when the early stage of question asking (here: the number of asked questions is less than 10). Based on the observed performance improvements when initializing the model using offline data, we conclude that using offline initialization is highly beneficial.

4.4.6 **RQ3.5** Online User Study

In this research question we first want to examine the assumptions made in this chapter. In particular, we first want to understand how many questions users are willing to answer, how well do they answer them, and how is their experience with the system. We collected 489 conversations made between our Qrec system and 21 crowd workers on 33 target items. From the collected data, we observe that crowd workers answered an average number of 15 questions per target item in the system (with the median being 12). Further, in the exit questionnaire, 71.4% of the crowd workers declare that they are willing to answer between 10 and 20 questions. Despite a median time of 5 seconds to answer a question, in the exit questionnaire, 95.2% of the crowd workers indicate that the system's questions were easy to answer. From the results we collected, most of the crowd workers think the conversational system is helpful and they will use it in the future. In particular, 81% of crowd workers found the experience positive, 14.3% neutral, and 4.7% negative. Last but not least, the crowd workers provided the correct answers to the system's question 95% of the time, they were not sure about their answers 3.5% of the time, and they gave the wrong answers (i.e., their answers disagreed with the description of the product) 1.5% of the time.

The second important question is how well the system performed. We measured performance after 5, 10, 15, and 20 queries asked (for those conversations that had this number of questions), as well as the performance when the crowd worker indicated that she wanted to stop. The results are shown in Table 4.5, and are in agreement with the Qrec results of Table 4.2.

4.5 Conclusions and Discussion

In this chapter, we propose a novel question-based recommendation method, Qrec, which directly queries users on the automatically extracted entities in relevant documents. Our model is initialized offline by our proposed matrix factorization model QMF and updates the user and item latent factors online by incorporating the modeling of the user answer for the selected question. Meanwhile, our model tracks the user belief and learns a policy to select the best question sequence to ask. Experiments on the Amazon product dataset demonstrate that the effectiveness of the Qrec model compared to existing baselines.

In this chapter, the questions asked to users are based on the presence or absence of entities in the target items, following past work. A richer type of questions could be constructed by using other sources such as categories, keywords, labelled topics [221, 223], structural item properties, and domain-specific informative terms. Also, we ignore the fact that entities may be semantically related to the target item even though they are not contained lexically in the item documents. Further, we leave the number of questions asked as a parameter to be predefined instead of algorithmically decided. Our work uses a stand-alone algorithm that learns the informativeness of questions to ask based on GBS. One can also use other techniques (e.g., reinforcement learning) to learn the optimal question asking strategy, or incorporate more factors, e.g., the relatedness and importance level of different informative terms, to extend the work. Still, the user may change their target item during the interaction with the system [140]. Theoretically our method is able to deal with this kind of situation, with new answers received gradually for the new target item. Last, we conduct a small user study, however a larger and in-situ user study by intervening at the interface of a commercial recommender system would be more informative. We leave all these as future work.

In this chapter, to answer **RQ3**, we have proposed a question-based approach to assist recommendation and validate the effectiveness of asking CQs by the proposed question-based approach. On top of Chapter 2 and Chapter 3, we demonstrate that our question asking strategy over entities can be incorporated with traditional recommendation techniques like matrix factorization. Next, we focus on the evaluation of CQ-based systems via user studies.

Part II

Evaluation

5 A User Study on Question-based Product Search

In this chapter, we focus on a user study on clarifying questions (CQs) in product search systems, to quantify and examine some assumptions in existing question-based product search systems. The ideal users would be ones who want to buy a number of products on an online shopping platform and now converse with our system embedded in the platform to find their target products. In the absence of such a user base and commercial system we simulate users with crowd workers via a crowdsourcing platform. We answer the following research question:

RQ4 To what extent can users answer CQs of question-based product search systems?

5.1 Introduction

One of the key components of conversational search and recommender systems [213, 219] is the construction and selection of good CQs to gather item information from users in a searchable repository. Most current studies either collect and learn from human-to-human conversations [27, 113, 170], or create a pool of questions on the basis of some "anchor" text (e.g., item aspects [213], entities [219, 224, 225], grounding text [143]) that characterizes the searchable items themselves. Although the aforementioned publications have demonstrated success in helping systems better understand users, most of them evaluate algorithms by means of simulations that assume that users are willing to provide answers to as many questions as the system generates, and that users can always answer the questions correctly, i.e., they always know what the target item should look like in its finest details. On the basis of such assumptions, their evaluations (e.g., Zhang et al. [213], Zou and Kanoulas [219], Zou et al. [225]) focus on whether the system can place the target item at a high ranking position. To the best of our knowledge, there is no empirical validation of whether and to what extent users can respond to these questions, and the usefulness perceived by users while interacting with the system.

In this chapter we conduct a user study by deploying an online question-based product search system to answer the following subquestions decomposed from **RQ4**:

This chapter was published as [226].

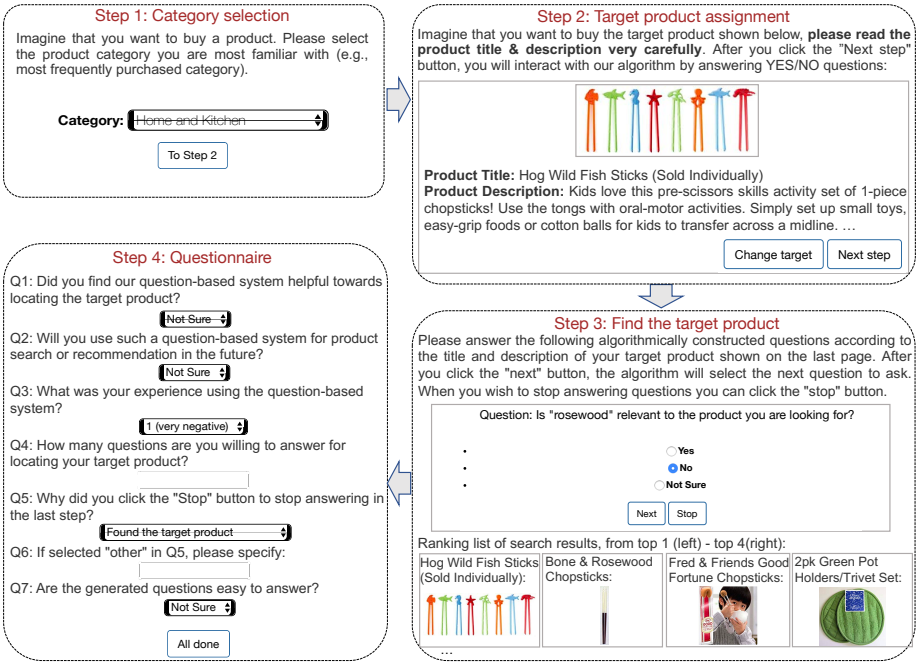


Figure 5.1: Question-based product search system architecture and main UI pages.

- To what extent are users willing to engage with a question-based product search system?
- To what extent can users provide correct answers to the generated questions?
- How useful do users perceive a question-based product search system to be while interacting with it?

We believe that answering these research questions can help the community design better evaluation frameworks and more robust question-based product search systems.

5.2 Study Design

In our study, given the absence of commercial systems and users who want to buy a number of products on an online commercial system and now converse with our system embedded in the commercial system to find their target products, we simulate users with crowd workers via a crowdsourcing platform, like other publications [134, 225]. The crowd workers interact with our question-based product search system in the domain of online retail. The crowd worker is answering questions prompted by the system with a "Yes", a "No" or a "Not Sure", in order to find a target product to buy, following the protocol:

(1) Complete a demographic survey inquiring on gender, age, career field, English skills and online shopping experience.

(2) Select a product category that they are familiar with.

(3) Read and select one target product from the assigned products (from the selected product category) that they were presented.

(4) Provide answers for a sequence of system's questions asked, and click on the "Stop" button to stop answering questions at any time when they want to stop.

(5) Answer a questionnaire regarding their perception and experience.

The architecture of our system is shown in Figure 5.1, with the crowd worker going through 4 steps. We also offer crowd workers detailed instructions for the study, as shown in Appendix A.

Step 1: Category selection. In this step, the crowd workers select an Amazon category[1] that they feel most familiar with to fit their interests, e.g., a category from which they have purchased products before.

Step 2: Target product assignment. We randomly assign a target product to the crowd worker from the selected category. The crowd worker is requested to read the title and the description of the product carefully. A picture of the product is also provided. This simulates a use case in which the crowd worker really knows what she is looking for, as opposed to an exploratory use case. If the crowd worker is not familiar with the target product the crowd worker can request a new product.

Step 3: Find the target product. After the crowd worker indicates that the conversation with the system can start, the target product disappears from the screen and the system selects a question to ask to the crowd worker. The crowd worker needs to provide an answer on the basis of the target product information she read in the previous step. Once the crowd worker answers the question, a 4-by-4 grid of pictures of the top sixteen ranked products is shown to the crowd worker, along with the next CQ. The crowd worker can stop answering questions at any time when she wants to stop during her interaction with the system. To select what CQ to ask, a state-of-the-art algorithm [219] is deployed to first extract important entities from each product description (e.g., product aspects) and construct questions in the form of "Is *[entity]* relevant to the product you are looking for?". Then, it selects to ask the information-theoretically optimal question, that is, the question that best splits the probability mass of predicted user preferences over items closest to two halves, and updates this predicted preference on the basis of the user's answer [219]. In this chapter, we update the predicted preference using the correct answer, i.e., the answer that agrees with the description of the product, independent of the crowd worker's answer. In other words, we study the user behavior under a perfect system from an information theoretical point of view, leading to a best-case analysis and conclusions.

[1]Categories and dataset we used: http://jmcauley.ucsd.edu/data/amazon/links.html.

Step 4: Questionnaire. In this step crowd workers are asked a number of questions about their experience with the system for further analysis.

5.3 Experiments and Analysis

5.3.1 Research Questions

Our research questions revolve around the user engagement and perceived value of the system:

RQ4.1 Are users willing to answer the CQs, how many of them, when do they stop and why, and how fast do they provide the answers?

RQ4.2 To what extent can users provide correct answers given a target product, and what factors affect this?

RQ4.3 How useful do users find the CQs, what is their overall experience, and how likely is it to use such a system in the future?

5.3.2 Participants

Prior to the actual study, we ran a pilot study with a small number of crowd workers, in a controlled environment, and iterated over the experimental design, and the instructions until no issues or concerns were reported. For the actual study 53 crowd workers located in the USA were recruited as participants through Amazon Mechanical Turk and 1025 conversations were collected. The participants were of varying gender, age, career field, English skills and online shopping experience. In particular, gender: 34 male, 19 female; age: 2 in 18–23, 8 in 23–27, 14 in 27–35, 29 older than 35 years old; career field: 22 in science, computers & technology, 8 in management, business & finance, 7 in hospitality, tourism, & the service industry, 3 in education and social services, 2 in arts and communications, 2 in trades and transportation, 9 did not specify their career field; English skills: all were native speakers; online shopping experience: 44 were mostly shopping online, 9 did online shopping once or twice per year. Participants were paid 2.5 dollars to complete the study. Also, we only engaged Master Workers,[2] filtered out those participants who spent less than 3 seconds on reading the product title and descriptions, and participants who gave random answers ($\sim$ 50% correct/wrong), for quality control.

5.3.3 **RQ4.1** User Willingness to Answer Questions

In **RQ4.1**, we first investigate whether participants are willing to answer the question-based product search system's questions and how many of them, both by observing the actual number of questions participants answered when interacting with the question-based product search system and what they declared at the exit questionnaire. The results in Figure 5.2 show that participants answer a minimum of 2 and a maximum of 48 questions. The average number of questions answered per target product is 11.4,

[2]High performing workers identified by Mechanical Turk who have demonstrated excellence across a wide range of tasks.

Table 5.1: The % of correct, "not sure", and incorrect answers.

Correct	73.1%	Not sure	9.6%	Incorrect	17.3%

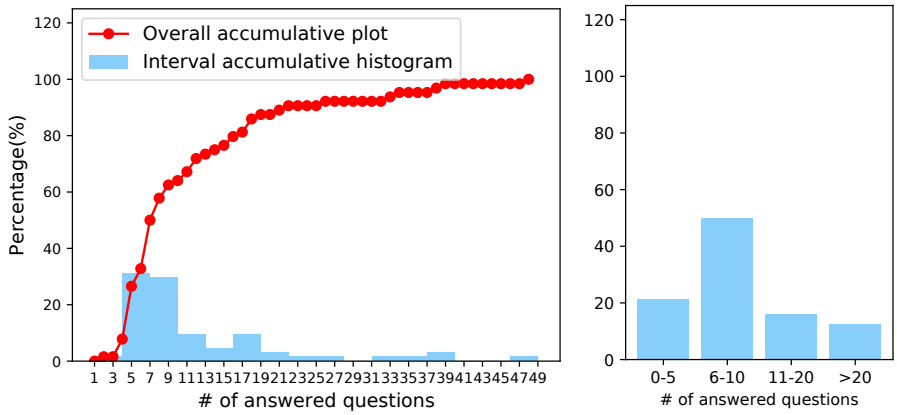
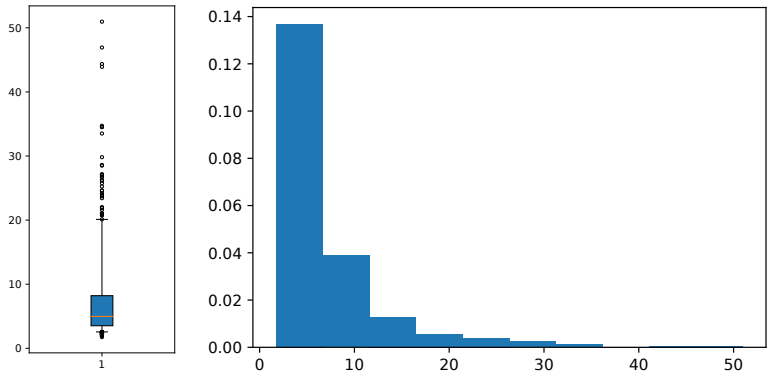


Figure 5.2: The number of questions the participants actually answered in the question-based product search system (left) and declared in the exit questionnaire (right). In the question-based product search system, the average number of answered questions per product is 11.4, and 70.3% of participants answered 4–12 questions per product. In the questionnaire, 50% participants are willing to answer 6–10 questions.

the median is 7, and 70.3% of participants answered 4–12 questions per product, while at the exit questionnaire about 50% of the participants declare that they are willing to answer 6–10 questions. Further, we explore why participants stop answering questions. Participants could select one out of six answers during the exit questionnaire: "The target product was found", "A similar product was found", "I got tired of answering questions", "I could not answer the questions", "The questions asked were irrelevant", and "Other". The results under the oracle condition show that while a small percentage of participants stop due to fatigue (14%) or due to irrelevant questions being asked (7%), the big majority of participants (77%) stop because they located the target product. We then analyze how fast the participants are in answering questions. Figure 5.3a shows a box-plot of the time spent per answer, while 5.3b better demonstrated the distribution. From the results, we observe that the minimum time for answering one question is 1.75 seconds, the average time is 7.1 seconds, and the median time is 4.98 seconds. 86.5% of the participants spent from 1.75s to 11.59s. Despite a median time of 5 seconds to answer a question, in the exit questionnaire 98% of the participants indicate that the system's questions are easy to answer.

5.3.4 **RQ4.2** User Answers Noise

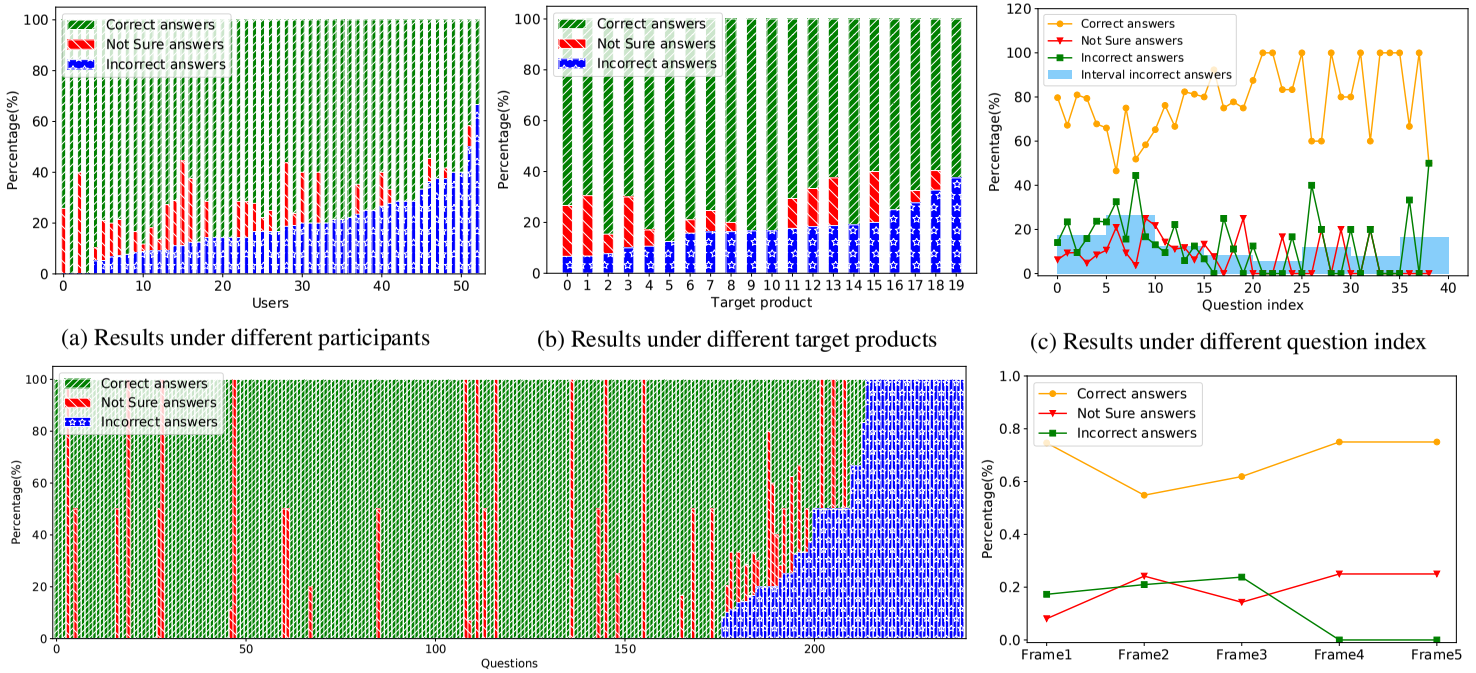
For **RQ4.2**, we first explore to what extent participants can provide correct answers. As one can observe in Table 5.1, participants provide correct answers 73.1% of the time, they are not sure 9.6% of the time, and they are wrong 17.3% of the time. We



(a) Box plot (b) Histogram (probability density) of time spent per question

Figure 5.3: Time spent per question by a participant in order to provide an answer. The average time for answering one question is 7.1 seconds.

then explore what features affect the percentage of incorrect answers. In particular, we first investigate whether the percentage of incorrect answers is different for different participants. The results in Figure 5.4a show the percentages of correct, "not sure", and incorrect answers vary across participants. Further, we explore whether the percentage of correct answers differs across target products. The results are shown in Figure 5.4b. We conclude that the percentage of incorrect answers varies across target products, but not as much as it varies across participants. The percentages of correct, "not sure", and incorrect answers for different questions asked by the system are shown in Figure 5.4d. Here we observe some dramatic differences across questions, with a smaller subset of questions receiving almost always incorrect answers. This might be because some questions are more ambiguous than others. This finding suggests improvements of question-based product search systems in multiple directions. For instance, one can try to improve the question pool by considering different question characteristics, or one could develop question selection strategies that also account for the chance of a user providing the wrong answer. Further, we explore whether the percentage of incorrect answers is correlated to the question index (i.e. i-th question), or whether it remains stable throughout the conversation. The results are shown in Figure 5.4c, where the lines show the average percentages of correct, "not sure", and incorrect answers as a function of the question index within the conversation, while the histogram shows the average incorrect answer percentages of a sliding window. The percentages fluctuate, but in principle they remain at similar levels throughout the conversation. Last, we explore whether the percentage of incorrect answers is correlated with the time spent to give the answers. The results within different time intervals are shown in Figure 5.4e. We divide the time spent per question (1.75s – 50.96s) into 5 equal non-overlapping buckets (or frames). We see that the percentage of incorrect answers decreases with more time spent. Also, we calculate the time spent when participants are giving a correct answer, a "not sure" answer, and an incorrect answer, with the averages being 6.59s, 10.81s, and 7.12s respectively, and the median 4.65s, 8.20s, 5.06s respectively. This suggests



(a) Results under different participants

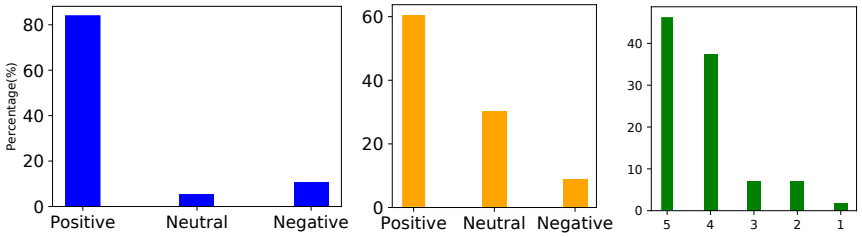
(b) Results under different target products

(c) Results under different question index

(d) Results under different questions

(e) Results for different time windows

Figure 5.4: The percentage of correct answers, “not sure” answers, which cannot be classified, and incorrect answers. (a) The % varies per participant; (b) the % varies across different target products; (c) the % remains stable through out the conversation; (d) the % varies per question, with only few questions receiving most of the incorrect answers; (e) the % of incorrect answers decreases as more time is spent.



(a) (b) (c)

Figure 5.5: User perceived helpfulness. (a) Is the question-based product search system helpful; (b) will you use the question-based product search system in the future; (c) ratings of participant experience. Most participants are positive towards our question-based product search system.

that participants usually spent more time when they are not sure about the answers, but almost the same time when they are right or wrong about a question.

5.3.5 **RQ4.3** User Perceived Helpfulness

Regarding **RQ4.3**, we explore how useful participants perceive a question-based product search system to be while interacting with it. We ask participants (a) whether they think the question-based product search system is helpful for locating their target products, (b) whether they will use such a question-based product search system in the future, and (c) what their experience rating is for the question-based product search system, ranging from 1 (very negative) to 5 (very positive). The results using oracle answers are shown in Figure 5.5, in the three plots respectively. From the results we collected, most participants think the question-based product search system is helpful and they will use it in the future. Specifically, 83.9% of participants are positive about the helpfulness, 5.4% are neutral, and 10.7% are negative. Further, 60.7% of participants are positive about using such a system in the future, 30.4% of participants are neutral, and 8.9% of participants are negative. Regarding ratings, the results show 46.5% of 5-star ratings, 37.5% of 4-star ratings, 7.1% of 3-star ratings, 7.1% of 2-star ratings, and 1.8% of 1-star ratings. 84% of the participants gave a rating at least as high as a 4.

5.4 Conclusion and Discussion

In this chapter we conduct an empirical study using a question-based product search system to gain insight into user behavior and interaction with such systems. We deploy a state-of-the-art question-based product search system online and collect interactive log data and questionnaire data for analysis. We find that participants are willing to answer a certain number of the system generated questions and stop answering questions when they find the target product, only if the questions are relevant and well-selected. While participants are able to answer these questions effectively, they also provide incorrect answers at a rate of about 17%, which varies across participants, products and question characteristics. Last, most participants are positive towards question-based product

search systems, and think that these systems help them towards achieving their goals of locating target products. The take-home message, if there is one, is that current research should drop the assumption that users are happy to answer as many questions as the system generates and that all questions are answered correctly.

One limitation of this chapter is the isolated clarifying-based environment of the study. A more realistic experiment would require CQs to be embedded in an existing environment, where the user is enabled to not only answer questions, but also reformulate her query or filter results by selecting pre-defined item attributes, and browse the results to the preferred depth. Also a mixed-initiative approach under which a system switches from asking questions, to understanding user searches, and combining the two is worth studying. A further limitation of this chapter is the fact that this was not an in-situ experiment by involving real users but a simulation of a use case of a question-based product search system by involving crowd workers. Hence, the findings are as good as our simulation of a user looking for a target product. Other factors, such as question quality, question format, and noisy answers, may affect the results, and studying therefore these factors in an A/B testing experiment would be beneficial. We leave all these as future work.

In this chapter, to answer **RQ4**, we have conducted a user study on question-based product search, and investigate the extent of user willingness and ability to answer CQs. To better understand user interactions for question-based systems, we also explore user interactions for the other question-based scenario, i.e., question-based web search, in the following chapter.

6 A User Study on Question-based Web Search

In this chapter, beside previous chapter focusing on question-based product search, we aim to better understand how users interact with clarifying questions (CQs) for another question-based scenario, i.e., question-based web search. This chapter examines the following research question:

RQ5 How do users interact with CQs in web search?

6.1 Introduction

To initiate the search process, users formulate queries that are meant to capture their information needs [176]. These search queries, however, are often short, ambiguous, incomplete, faceted, or misinterpreted by search algorithms due to lack of user's context [206]. For search systems to better respond to users' queries researchers have introduced techniques such as personalization [172], context awareness [197] and result diversification [179]. While effective in web search, personalization and contextualization leverage user data that is not always accessible; diversification requires the user to scan multiple result snippets before finding the right result, which can lead to frustration [56]. Also, it is impractical for users on limited bandwidth interfaces to scan multiple results [206]. Alternatively, researchers have augmented search functionality by allowing search engines to ask CQs as a step towards better understanding users' information needs [5, 206], context, and preferences [146].

Asking CQs has gained interest within the information retrieval community due to the popularity of conversational information seeking systems. While building systems capable of having mixed-initiative interactions with users has been a long-standing goal [15, 60, 144], we have observed notable developments and successes in this area only recently [5, 6, 79, 191, 206, 219, 225]. The significance and effectiveness of CQs has been recongnized for broad use cases such as product search [213, 219], preference elicitation for recommendation [165, 225], information-seeking conversations [5, 79, 104], and web search [207]. These recent publications showcase the effectiveness of CQs for system performance, yet the impact of asking CQs on users is rather unknown.

This chapter is currently submitted as [227].

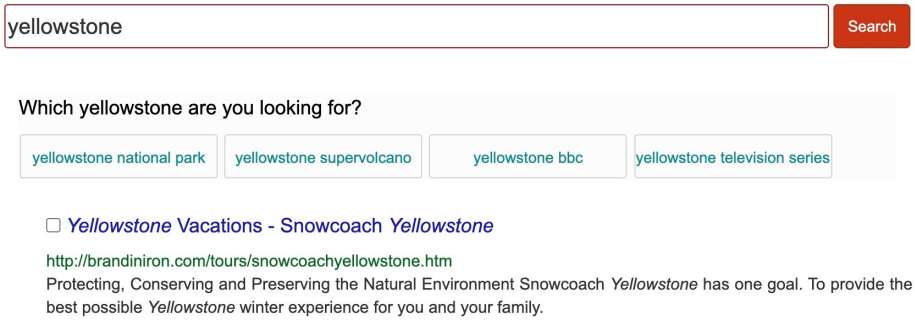


Figure 6.1: Search engine UI of our CQ-based web search system.

Findings of previous work indicate that users enjoy voice query clarification even though it delays the system's response [97]. Also, we know that different CQ templates, attributes of candidate answers, and query properties affect user engagement rate [208]. However, the impact of different quality level of CQs on user's search performance, and the effect of user perception on search clarification remain unstudied. For example, displaying a clarification pane on top of the Search Engine Results Pages (SERP) or interrupting users in a conversation bears an unknown effect in terms of cost and benefit on users. It can be beneficial to guide a user through their search by asking one or multiple CQs, but low quality CQs may come with a high risk of frustrating users.

In this chapter, we investigate (a) the effect of asking different quality of CQs on users' search behavior, ability to find relevant information and satisfaction, and (b) factors regarding users' background, perception intrinsic to the search task, or circumstances that affect the need of users to engage with CQs. To this end, we conducted a user study involving 106 participants during which we ask them to complete a web search task. In particular, we simulate various conditions that a user and a system would encounter and study how different system decisions would affect users' behavior and how user decisions would affect system effectiveness. By design, the search tasks span various topics and levels of difficulty, as well as CQ quality categories. Two groups of users participated in the study, with one group completing search tasks using a plain search interface, and the other group using a search clarification interface, designed to resemble Bing's[1] clarification pane [206]. As shown in the sample search interface in Figure 6.1, the clarification pane consists of a CQ and the corresponding suggested answers displayed below the query input. Analysis of collected implicit and explicit user data allows us to look into users' behavior and satisfaction, within and across the two groups.

With our work, we answer three subquestions decomposed from **RQ5**:

> **RQ5.1** To what extent does asking CQs affect users' search behavior and satisfaction? Are users affected by being asked high quality vs. low quality CQs in a search session?

To address **RQ5.1**, we present users with different categories of CQs with varied

[1] http://www.bing.com

quality and compare behavioral measures capturing interaction and performance such as querying, mouse movement and bookmarking. We also investigate how much engaging with different quality categories of CQs affects user satisfaction. Moreover, we hypothesize that the impact of CQs on users' performance spans further to the next SERP and roots in the whole search session. As such, we analyze how users are affected not only immediately after interacting with CQs (query-level), but also in the whole session as they strive to complete a search task (session-level). To deepen our understanding of users' interactions with CQs, we also seek to answer:

RQ5.2 How much do user background and task perception affect their interactions with CQs?

To address **RQ5.2**, we analyze responses to pre/post task questionnaires and user demographics information. With **RQ5.1**, we examine how user behavior is affected after engaging with the CQs; here we investigate the extent to which user's task perception such as expected difficulty and prior knowledge affects their willingness to interact with CQs. With our last research question, we draw attention to user engagement with CQs under different circumstances, trying to answer:

RQ5.3 How do users interact with CQs under various circumstances?

To address **RQ5.3**, we calculate user engagement metrics with the CQ pane (e.g., click through rate and cursor hovering), and study how factors like CQ quality categories, task types and SERP quality affect user engagement with CQs. As generally users answer CQs (in the form of clicking on an answer) [207, 226], it is also critical to study how other engagement metrics such as mouse movement differ among CQs of different quality and type.

Our results indicate that (a) when users engage with high quality CQ panes, the interaction, performance and satisfaction increases, compared to a search engine that does not offer such an option; however when the CQs are of low or mid quality they actually negatively affect all measures, even if they are presented to the user and the user does not engage with them; (b) users' expected and perceived difficulty of the search task influenced the degree of their engagement with CQ, while less experienced users made a wrong use of the CQs, clicking more on irrelevant answers; and (c) users' degree and quality of engagement with CQ panes are affected by factors such as SERP quality, diversity, screen size and decrease as a search session evolves. As asking CQs is a necessary step towards developing mixed-initiative conversational search systems [144, 207], we believe that our findings can help towards this direction.

6.2 Related Work

Asking CQs has shown great potential to enhance functionality of a number of applications, such as search [154, 213, 219], recommender systems [165, 225], information-seeking conversations [5, 79, 191, 206], and dialogue systems [49, 168, 201]. Four decades ago, Belkin et al. [15] explored early mixed-initiative systems by offering users choices in a search session and reflected the significance of mixed-initiative systems. Recently, Zamani et al. [206] proposed a neural approach to generate CQs. Hashemi

et al. [79] used CQs to enrich representation learning in information-seeking conversations. The importance of CQs for conversational search and recommender systems has also been highlighted by Radlinski and Craswell [144]. Zhang et al. [213] presented a unified approach for conversational search and recommendation by asking questions over item "aspects" extracted from user reviews. Instead of item "aspects", Zou et al. [219, 225] construct CQs based on extracted informative terms, for recommendation and product search, respectively. Asking CQs about different item attributes is also applied to improve conversational recommender systems and dialog systems [6, 218]. Given that asking CQs is a prominent area of study, we have recently seen an influx of datasets and challenges facilitating research in this area. Notable examples include the Qulac dataset [5], the MIMICS dataset [207], and the Conversational AI challenge [6]. These datasets and challenges enable researchers to train systems and evaluate them on tasks related to CQs. Existing work on CQs mainly focuses on model and representation learning, and the construction of datasets; instead, we aim to study the underlying mechanism of user interactions with search systems using CQs, offering insights into the design of these models.

Research discussing empirical studies examining CQs is broad, from the use of CQs on community question answering sites like Stack Exchange, where answerers ask CQs to askers to better comprehend information requests [20] to the challenges of CQs for entity disambiguation [33]. Vtyurina et al. [184] compare three different conversational search systems: humans, assistants, and wizards; Kiesel et al. [97] study the effect of query clarification over voice on users' satisfaction and found that language proficiency affects users' satisfaction; Trippas et al. [178] study the impact of voice query clarification on user interaction, and found that the user query and the average time on task became longer as the task complexity increases.

More recently, Krasakis et al. [104] analyzed the effect of CQs on document ranking. Zou et al. [226] empirically quantified and validated user willingness and the extent of providing correct answers to CQs in existing question-based product search systems. Different from their work which mainly validates certain assumptions made by existing CQs-based models, we study user behavior and engagement with different quality categories of CQs for search clarification.

Zamani et al. [206] conducted a user study showing that asking CQs is in principle beneficial. They constructed a taxonomy of clarifications for open-domain search queries with the purpose of developing CQ templates. Based on their previous work, Zamani et al. [208] conducted a large-scale in-situ study, analyzed clarification panes for millions of queries, and developed representation learning methods to re-rank clarification panes. In particular, they analyzed the click rate received on CQ panes as a function of search query properties (e.g., query length), question template types (on the basis of their template taxonomy), and answers attributes. Also, they analyzed the impact of clarification on proxies of user dissatisfaction. Our work is complementary to the work by Zamani et al. [208]. We conduct a smaller scale but controlled laboratory user study. This allows us to control certain variables, e.g., the quality of the CQ panes or the relevance of the results, and collect explicit user information (e.g., via questionnaires), and user feedback (e.g., bookmarks). Further, we focus our analysis on user search performance, behavior and satisfaction, at the query and session level, as well as the need for user engagement with CQs.

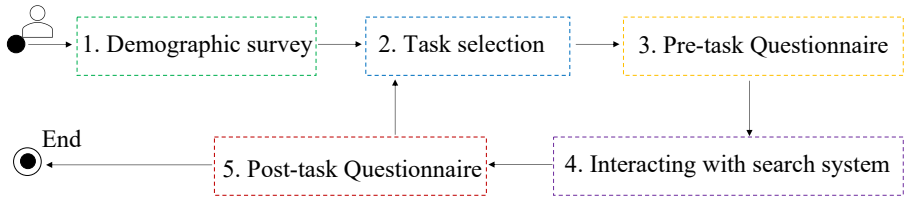


Figure 6.2: Study protocol.

6.3 Method

We present here our study protocol, design, and participants.

6.3.1 Protocol

To answer our research questions, we conducted a user study in order to capture implicit user behavior and explicit user feedback in varied conditions. Following the protocol in Figure 6.2, we ask study participants to:

(1) Complete a demographic survey.

(2) Read and select one search task from the list that they were presented.

(3) Answer a pre-task questionnaire regarding their perception and opinions for the selected task.

(4) Submit a search query and find relevant information and bookmark it. During their session they can choose to answer a CQ (if shown to them). However, it was not a requirement to answer or engage with the CQs.

(5) Click on the "Finish" button, as soon as they are confident that they have found relevant information.

(6) Answer a post-task questionnaire regarding their perception and experience.

We offer participants detailed instructions for the study and clarify its goal, as shown in Appendix B. We demonstrate how our augmented search interface works in a short video and stress that CQs are meant to support their search but that they are not always related and useful; we urge them to proceed with web search tasks as they normally would and take advantage of CQs only if they find them useful during their sessions.

We did not collect any private data from the users that can be used to breach their privacy. Furthermore, our study was approved by the ethics committee of the institute and we specified to the participants that their data is securely encrypted and stored and that they can opt out at any point of the study.

6.3.2 Study Design

In this section we offer insights on design decisions that constitute the cornerstone of our study.

Table 6.1: Task description. Topic id is represented by original topic id followed by subtopic id in parenthesis in TREC Web Track.

Task id	Topic id (Web Track)	Task Categories
T1	133(1)	fact-finding & faceted
T2	197(1)	fact-finding & faceted
T3	52(6)	information gathering & ambiguous
T4	60(1)	information gathering & ambiguous
T5	200(3)	information gathering & faceted

Search System Our search system is designed to mimic the commercial search system Bing [206], including the CQs pane embedded on the top of the search page–the search UI is shown in Figure 6.1.[2] In the search page, users need to think of a query based on the task they selected, enter the query in the search box and wait for results together with CQs (if any). Then users are able to (a) browse the results; (b) reformulate their query and repeat the search as many times as they wish; (c) answer the CQs; (d) click on the results that they find interesting and would like to know more about, and bookmark the ones they think are relevant for the task's information need. The retrieval model for the result page is based on Chatnoir [141] for the ClueWeb09 Corpus[3] (BM25-based), and duplicate and spam results[4] are removed. CQs and corresponding answers for each task are from the CQs pool (Section 6.3.2), with each answer corresponding to a reformulated query for the next turn. For each task, the CQ to be shown to the user is randomly selected from multiple CQs for that task. Each CQ has at most five answers following the Bing setting [208]. The answers to the corresponding CQ also appear in a random order in the CQs pane each time.

Search Tasks Different search tasks may influence information-seeking behavior [194]. Hence, we constructed five search tasks derived from the Text Retrieval Conference (TREC) Web Track 2009 - 2012.[5] To motivate the search session, the users are guided to read through all the task descriptions and select the task that they feel most comfortable with. To make clear to the users what is the task they are expected to complete, we expand the task description using Simulated Work Task Situations [19], which create a task scenario that offers participants with a search context and a basis for relevance judgments. See below for a sample task description:

> Imagine you are flying next week from the Ontario airport, located in California. Since this is your first time flying out from this airport, you are thinking of gathering some information about its facilities and services.

[2] We allowed participants to report system interface issues; none was reported.

[3] https://www.chatnoir.eu

[4] Spam filtering is performed by applying Waterloo Spam Ranking for the ClueWeb09 Dataset, which was typically applied for TREC Web Track 2009 submissions.

[5] https://trec.nist.gov/data/webmain.html

We categorize each task according to its type: fact finding or information gathering [75]. The former are simple tasks looking for specific facts, files, or pieces of information; the latter involves collecting information often from various sources in order to make a decision, write a report, or complete a project. Tasks are also categorized based on types defined by TREC: faceted and ambiguous (see Table 6.1). Study participants could complete several tasks among the ones provided. Prior to the experiment all tasks were pilot-tested until no issues were reported. Post-hoc analysis of the distribution of selected tasks during the full study showed no obvious preference for any task in either category, and most users (91.2%) said that the tasks were very clear.

CQs & Candidate Answers To study the mechanism of search clarifications, our CQ-based web search system depends on a pool of CQs and candidate answers. We first build a CQ taxonomy capturing different quality categories of CQs based on their relatedness and usefulness for the search process. Then, two expert annotators generated CQs and candidate answers for each task following this proposed taxonomy. They reviewed the CQs and candidate answers together after the generation. In case of disagreement, the annotators would discuss and agree on a common decision. To inform taxonomy design, we conducted a survey to ask users about factors that would lead them to interact with CQs. Based on $\sim$ 200 collected responses, most users indicated 'related question asked' (33.5%) and 'useful question asked' (21%); the latter aligns with the usefulness metric assessing the follow-up question suggestion in web search by Rosset et al. [154]. To refine our taxonomy, we also looked at existing CQs literature [154, 206, 208] and public CQs datasets. The question taxonomy includes three main categories: **(C1)** off-topic, unrelated CQs; **(C2)** related but not useful CQs, e.g., duplicate question with user query or a related question without useful answers; **(C3)** related and useful CQs. To make the development of CQs easier we further defined two subcategories for C3 according to two fundamental purposes, consisting of CQs **C3(i)** related and useful for specific/faceted details, e.g., a question asking for a faceted attribute; **C3(ii)** related and useful for disambiguation, e.g., query "apple" for fruit or Apple brand. Category C3 refers to good quality CQs that bring real value to the user, which could be a way of providing new information, a next step to complete a task, or exploratory options on the task. The general idea in creating this taxonomy is to cover a broad variety of quality categories of CQs and investigate the potential mechanism of search clarification under these quality categories. For example, CQs that are off-topic or useless may provoke user dissatisfaction and cause users to leave the session [191], whereas related and useful CQs can help users [226]. The taxonomy and sample CQs are shown in Table 6.2.

Questionnaires We present participants with a set of questionnaires to gather explicit feedback. We first show participants *demographic questions* eliciting information pertaining to their gender, age, career field, English language proficiency, and educational background. Responses to these questions help us understand users and whether their background influences interaction with CQs. Moreover, before and after completing each task, we ask participants to complete short questionnaires. From the *pre-task questionnaire*, we collect participants' perception and their opinions regarding their chosen

Table 6.2: CQs taxonomy and examples. C1, C2 and C3(i) from the task "scientific name of Idaho State flower;" C3(ii) from "movie named AVP".

Taxonomy	Description	Examples of questions & answers
C1	Off-topic, unrelated CQs	Q: What do you want to know about the Idaho state flag? A: 1. Year adopted; 2. Pictures; 3. History; 4. Designer; 5. Colors.
C2	Related but not useful CQs	Q: What do you want to know about the Idaho State flower? A: 1. Growing seasons; 2. Growing conditions; 3. History; 4. Color; 5. Year adopted.
C3(i)	Related and useful CQs for specific/faceted details	Q: What do you want to know about the Idaho State flower? A: 1. Growing seasons; 2. Growing conditions; 3. history; 4. Color; 5. Scientific name.
C3(ii)	Related and useful CQs for disambiguation	Q: Which AVP are you interested in? A: 1. AVP program; 2. AVP comapany; 3. AVP association; 4. AVP airport; 5. AVP movie.

task, including their prior knowledge, expected task difficulty, perceived task clarity, task interest, distraction level, and search expertise. From the *post-task questionnaire*, we gather information related to participants' experience with the system, including its perceived helpfulness, attitudes towards future use of CQ-based systems, perceived task relevance, perceived task difficulty, and domain knowledge for the completed task. From responses to the aforementioned questionnaires, we gather users' opinions about the task and the system, which allow us to investigate the relationship between user interactions with CQs and user background, task perception, as well as user experience.

6.3.3 Participants

We recruited 106 individuals via email invitation (students and staff of two universities, one in Europe and one in the U.S.) who took part in our user study. Participants were on a voluntary basis. Recruited participants varied in:

- Gender: 39 females, 65 males, 2 non-binary;
- Age: 69 in 18-24, 26 in 25-34, 7 in 35-44, and 4 participants were older than 44 years old;
- Career field: 86 in Science, Computers & Technology, 3 in Education and Social Services, 3 in Health Care, 3 in Law and Law Enforcement, 2 in Management, Business & Finance, 1 in Architecture and Civil Engineering, and 8 did not specify;
- English language proficiency: 22 native, 51 proficient, and the remaining ones were beginners; and

Table 6.3: Statistics of collected data.

# users	106
# tasks	5
# search requests	1,334
# CQs	15
# CQs showing/hiding times	1,016/318
# CQs clicks	249
# users' bookmarks	1,942
# users' clicked results	705
# users' cursor hovering records	17,780
# users' page scrolling records	15,747
avg. # tasks per user	3.11
avg. # search requests per user	12.58
avg. # search requests per task	266.8
avg. # bookmarks per user	18.32
avg. # bookmarks per task	388.4

- Highest education level completed: 58 high school, 21 Bachelor's, 11 Master's, 4 Doctorate, 12 did not specify.

6.4 Results

In this section, we detail our analysis of data collected through our user study. Data statistics are shown in Table 6.3. Unless otherwise reported, for comparisons between two groups only, t-tests were used for analyses in this chapter; for comparisons between more than two groups one-way analysis of variance (ANOVA) and Least Significant Difference (LSD) post-hoc tests were used to control for Type I errors [11]. To ensure the *data quality*, we performed two quality checks and filtered out low quality participants: (a) we asked questions about the study instructions to ensure that participants have read it carefully and understood it; and (b) we measured the time participants spent reading the task descriptions and filtered out participants who spent less than 10 seconds (a minimum expected threshold for a trustworthy worker [77]). We did not filter users based on their interactions with the CQs.

6.4.1 **RQ5.1** Impact of CQs on Search Behavior and Satisfaction

In **RQ5.1**, we explore the impact of CQ quality categories on users' search behaviors and satisfaction. For search behavior we consider three types of measures [95]:

(a) *Interaction*–number of queries issued, number of query terms, number of SERP scrolls, number of SERP hovers, and number of SERP clicks;

(b) *Performance*–number of results marked relevant (# bookmarks), number of correct bookmarks (# hit), and SERP quality measured by nDCG@10 (Normalized Discounted Cumulative Gain from rank 1 to 10);

Table 6.4: Objective behavior measures by condition, i.e., clicking on an answer related to each CQ quality category. * and † denote significant difference with No CQs and C3, respectively (*/† p-value <0.05; **/†† p-value <0.01; ***/† † † p-value <0.001).

	No CQs	C1	C2	C3
# bookmarks /page	1.56(2.14)††	0.41(0.86)***† † †	1.06(1.54)† † †	2.22(2.30)**
# hits/page	0.73(1.21)† † †	0.07(0.25)***† † †	0.79(1.12)†	1.17(1.39)***
nDCG@10	0.27(0.30)† † †	0.12(0.19)**† † †	0.25(0.27)† † †	0.42(0.39)***
SERP scrolls	12.73(14.70)† † †	3.64(5.99)***† † †	7.63(9.85)*† † †	19.20(17.63)***
SERP hovers	12.93(13.39)† † †	6.55(6.78)*† † †	9.85(10.79)† † †	19.55(21.25)***
SERP clicks	0.55(1.15)†	0.09(0.47)*† † †	0.27(0.65)††	0.83(1.43)*
dwell time(s)	56.32(68.83)†	36.61(83.18)† † †	34.38(33.82)*† † †	73.87(65.41)*

(c) *Time spent*–dwell time on SERPs per query and the overall task time.

For satisfaction, we use explicit feedback collected via the post-task questionnaires:

(a) overall satisfaction rating;

(b) user perceived helpfulness;

(c) user attitude towards future use of CQ-based search systems.

Query-level Behaviors by CQ Quality Category. We start our analysis on query-level search behaviors. In Table 6.4 we report behavioral measures under different conditions, i.e., when clicking on a CQ pane of a certain quality (C1, C2, and C3), and when no CQ was shown to the user ('No CQs'). Throughout Table 6.4 we observe a similar behavior for all the metrics reported. When a user engages with a low quality CQ (C1) all metrics are low, and typically much lower than for a search interface that does not offer CQ panes. The metrics increase when the user engages with a mid quality CQ (C2), in which case all metrics are on par with the case of searching without CQs. All metrics significantly improve when the user engages with high quality CQs (C3).

Bookmark quality. From Table 6.4 we see that engaging with good quality CQs (C3) leads to a significant increase in the number of (correct) bookmarks, i.e., '# bookmarks/-page' and '# hits/page', compared to searching without the use of CQs; the opposite occurs when the user engages with CQs that belong to C1. The number of (correct) bookmarks also significantly increases from either C1 or C2 to C3. This indicates that high quality CQs help users find relevant information, while mid and low quality CQs have a negative impact.

SERP quality. SERP quality, measured by nDCG@10, significantly increases after clicking high quality CQs (C3) but significantly decreases after clicking low quality CQs (C1), compared to 'No CQs'. Compared to C3, engaging with C1 or C2 CQs significantly lowers the SERP quality.

SERP scrolls & hovers. Cursor movements like scrolling and hovering are valuable signals for inferring user behavior and preferences [89]. Thus, we investigate the impact

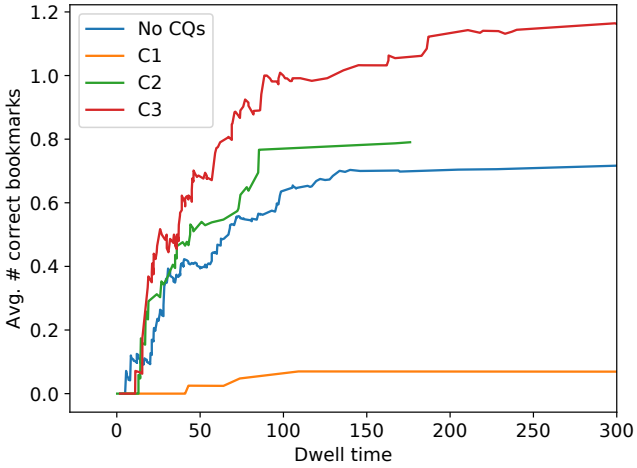


Figure 6.3: Dwell time vs. # correct bookmarks (# hits).

of different quality categories of CQs on SERP scrolls and hovers. We observe that the number of scrolling significantly increases after engaging with C3 compared to C1/C2/'No CQs'; it significantly decreases after clicking C2 or C1 CQs, compared to 'No CQs'. Further, we note that the number of hovers also significantly increases after clicking on C3 CQs compared to C1/C2/'No CQs'; it significantly decreases after clicking on C1 CQs compared to 'No CQs'. This indicates that users scan a SERP more extensively when they engage with a high quality CQ and less in case of mid and low quality CQ.

SERP clicks. The number of clicks on SERP results significantly increases after engaging with high quality CQs compared to mid and low quality or 'No CQs'; it also significantly decreases after engaging low quality CQs compared to 'No CQs'. This may be due to improved SERP quality and users see more interesting results on SERP after they engage with high quality CQs, but less interesting results after they click C1/C2 CQs.

Dwell time. Users spend significantly more time on SERPs after engaging with high quality CQs as opposed to C1/C2 CQs. This might be because users realize the information is not there and quickly move on after clicking C1/C2 CQs, and they pay more attention and attempt to find more results that they think are relevant (# bookmarks) after they click C3 CQs. Compared to 'No CQs', dwell time significantly increases after clicking C3 CQs, but it decreases after clicking C1 CQs (not significantly) or C2 CQs (significant, $p<0.05$).

Evolutionary trend of # hits with dwell time. To better understand the impact of different quality categories of CQs, we explore the evolutionary trend of the average number of hits with dwell time on the secondary page after users engage with a CQ pane (and hence a new query is submitted to the system), so as to consider trade-offs between CQs engagement and dwell time. From Figure 6.3, we see that the average number of hits increases for C1, C2, C3, and 'No CQs', i.e., more time spent leads to more relevant

information found. However, the growth rate greatly declines at later stages, which indicates the diminishing return of gain [10]. Further, clicking on high quality CQs (C3) always leads to more relevant information found with respect to the dwell time cost, whereas clicking low quality CQs (C1) results in a lower benefit than 'No CQs'.

Session-level Behaviors by CQ Quality Category. Besides examining the immediate effect that interacting with CQs has on user behavior, we explore the effect of CQs across the entire session. We report the sum values for different measures in the whole session in Table 6.5. By design, after each user's query CQs in our study were chosen at random (for the group of participants that viewed CQs). Therefore, sessions in which a single category of CQs appeared are rare. Hence, we split sessions into four categories, along two axes: (a) sessions w/o any C3 category CQs vs. sessions w/ C3 category CQs, and (b) sessions in which users engaged (i.e., clicked on an answer) with a CQ vs. sessions in which users did not engage with (i.e., skipped) the CQ panes. In principle, the results in Table 6.5 when the users click on high quality CQ panes follow those in Table 6.4. When users do not click on CQ panes (the last two columns), we still observe an interesting gap between high quality CQ panes vs. lower quality CQ panes, indicating that the quality of CQ panes has indirect effects on user behavior.

Bookmark quality. Similar to query-level trends, clicking an answer from a C3 category CQ pane significantly increases the number of correct bookmarks in the whole session compared to 'No CQs'. 'No CQs' sessions or sessions in which users clicked an answer for C1 or C2 CQs result in a comparable number of correct bookmarks. Further, viewing CQs of C1 or C2 with no clicks (i.e., w/o $C3^{-}$) results in a slightly higher number of correct bookmarks than clicking C1/C2 CQs (w/o C3), which indicates clicking a low quality or mid quality CQ is harmful.

SERP quality. Clicking C3 CQs in the session (w/ C3) significantly increases the SERP quality compared to 'No CQs' and skip C3 (w/$C3^{-}$).

SERP scrolls & hovers. Engaging with CQs of w/ C3 session significantly increases the SERP scrolls and hovers compared to w/o C3 and w/$C3^{-}$ session (skip C3).

SERP clicks. The number of clicks for SERP results significantly increases after clicking C3 CQs compared to clicking CQs of C1 or C2 (w/o C3 vs. w/C3).

Users' queries. Clicking on CQs of either w/o C3 or w/C3 significantly increases # query terms and session length. This indicates users tend to formulate significantly shorter queries by themselves ('No CQs') compared to automated reformulated queries by clicking CQs, revealing an advantage of CQs when long queries are needed.

Overall task time. Clicking CQs of w/ C3 slightly increases the mean dwell time per task whereas clicking CQs of w/o C3 slightly decreases it (not significantly different). This might be because users pay more attention and are able to locate more relevant results to bookmark (average # of bookmark per search session: 6.05 vs. 5.36). Skip C3 (w/ $C3^{-}$) significantly decreases dwell time compared to clicking C3 (w/ C3). In addition, we also find that the mean dwell time per task when clicking C3 CQs for information gathering tasks is higher than fact finding tasks (260.95s vs. 223.47s).

Table 6.5: Objective behavior measures by condition, i.e., viewing and clicking/not clicking on CQs in a session. *, † and § denote significant difference with No CQs, w/ C3 and w/o C3$^-$, respectively. (*/†/§ p-value <0.05; **/††/§§ p-value <0.01; ***/† † †/§§§ p-value <0.001).

		Click on Answer		**No Click on Answer**	
	No CQs	w/o C3	w/ C3	w/o C3$^-$	w/ C3$^-$
# bookmarks/session	5.36(2.47)	5.64(3.52)	6.05(3.18)	6.08(3.39)	6.10(2.93)
# hits/session	2.40(1.86)†	2.53(1.99)	3.17(2.45)*	2.66(2.21)	2.57(2.20)
nDCG@10	0.87(1.23)† † †	1.11(1.00)	1.42(1.29)***§§§	0.70(0.74)† † †	0.90(1.05)†
SERP scrolls	46.03(32.48)	38.03(29.69)†	55.67(43.13)§§	36.34(34.04)††	37.3(31.63)†
SERP hovers	42.90(30.36)† † †	43.14(29.20)††	65.15(48.80)*** §§§	41.06(32.70)† † †	39.37(23.48)† † †
SERP clicks	2.22(3.02)	0.96(1.74)†	2.34(3.19)	1.76(3.91)	2.7(3.41)
# queries/session	3.18(2.30)† † † §	4.50(2.13)*§§§	5.02(2.88)***§§§	2.28(1.28)*† † †	3.57(2.17)†† §
# query terms	11.43(12.19)† † † §	17.96(10.01)**§§§	19.24(12.70)***§§§	7.18(5.74)*† † †	11.3(7.77)† † †
task time(s)	204.75(173.20)§	188.32(187.84)	244.70(192.72)§§§	136.22(129.34)*† † †	170.85(128.32)†

Table 6.6: User satisfaction measures by condition, i.e., viewing and clicking/not clicking CQs in a session. Satisfaction shows means followed by standard deviations in parenthesis. *, † and § denote significant difference with No CQs, w/ C3 and w/o C3$^-$, respectively (*/†/§ p-value $<$0.05; **/††/§§ p-value $<$0.01; ***/† † †/§§§ p-value $<$0.001). Future use and helpfulness are represented by ratio of positive/negative ratings.

		Click on Answer		**No Click on Answer**	
	No CQs	w/o C3	w/ C3	w/o C3$^-$	w/ C3$^-$
satisfaction	2.88(0.99)†	3.14(1.06)	3.26(1.14)*§§	2.72(0.98)††	2.70(1.00)†
future use(%)	48.61/16.67	53.57/32.14	70.63/8.39	44.00/22.00	50.00/23.33
helpfulness(%)	57.14/28.57	34.62/53.85	59.86/29.58	16.67/77.78	13.33/73.33

Satisfaction by CQ quality category From Table 6.6, we see that user satisfaction significantly improves when users interact with C3 compared to 'No CQs' (w/ C3 vs. 'No CQs'); when they skip C3, user satisfaction decreases significantly compared to clicking C3 CQs (w/ C3$^-$ vs. w/ C3).

To gauge attitudes towards future usage of CQs-based systems and helpfulness, in the post-task questionnaires we inquired on users' positive, neutral, or negative rating. Based on the percentage of positive and negative ratings across groups, we note that adding C3 quality CQs in the session improves user attitudes towards future usage of CQs-based systems (w/ C3 vs. w/o C3: more positive ratio and less negative ratio for w/ C3). Users who actually engage with high quality CQs are more positive compared to those that do not engage (w/C3 vs. w/C3$^-$). For user perceived helpfulness, we see that most users in w/ C3 (59.86%) are positive, while users in w/o C3 are in principle negative (53.85% negative). Adding C3 quality CQs in the session also improves user perceived helpfulness (w/ C3 vs. w/o C3). Viewing CQs but not clicking them yields a much lower percentage of positive users and a much higher percentage of negative users (w/o C3$^-$ and w/C3$^-$ vs. 'No CQs').

Previous studies [90, 119] suggest that the last impression (query) within a session may have a stronger correlation with users' search satisfaction. With this in mind, we study whether there is an impact depending on when the interaction with CQs took place. Dividing each session into three segments (first query, in-between queries, and last query) [90], we indeed note the last impression effect: users are significantly more satisfied when user interaction with CQs occurs on the last query compared to the first query (p<0.05) or in-between (p<0.01) (average satisfaction score: 3.27, 3.04 and 3.62 for first, in-between, and last query, respectively).

6.4.2 **RQ5.2** Impact of User Background on CQs Interactions

In **RQ5.2**, we explore the impact of the user's background as measured by demographics and perception on the extent of interactions with CQs, as measured by CQ answer click through rate (answer CTR, i.e., total clicks divided by total showing times).

User Demographics User demographics like age, education, and gender are among the most important predictors of online information search behavior [192]. This mo-

tivates our study of intrinsic characteristics that may lead users to interact with CQs in their quest for information. In turn, we gain valuable insights into how to make decisions about showing CQs to different users.

Gender. Our analysis reveals a significantly higher answer CTR in the case of female users compared to male users (25.2% vs. 23.6%, $p<0.05$), and lower correct answer CTR (the number of correct answers clicked compared to total answer clicks) than male users (51.4% vs. 55.6%), pointing to information processing differences between females and males [98].

Language. Language proficiency affects interactions, with the decrease in proficiency (native speaker $\rightarrow$ proficient $\rightarrow$ beginner), leading to answer CTR on C3 to drop (67.5%, 63.6%, 56.1%). Beginners engage significantly less on C3 than native speakers ($p<0.05$), and proficient speakers ($p<0.01$). Also, overall answer CTR drops (26.3%, 25.7%, 21.8%) and correct answer CTR drops (56.8%, 55.6%, 50.6%) with the decrease in proficiency. This observation is in agreement with Kiesel et al. [97] for voice query clarification.

Education. Answer CTR on C3 grows among users with a higher education background (57.1%, 65.6%, 78.3%, 100% for high school, Bachelor's, Master's, and Doctorate's degrees, respectively). Users with Bachelor's degrees or above engage significantly more on C3 than users without Bachelor's degrees ($p<0.05$); same for users with Doctorate degrees ($p<0.001$).

Career field. As anticipated, computer science students, likely more well-versed on search literacy instruction, exhibited higher overall answer CTR with CQs compared to other careers (25.2% vs. 22.7%). While not significant, we notice a higher answer CTR on C3 (64.1% vs. 56.3%), and a higher correct rate for correct answer clicks (55.7% vs. 48.4%).

Age. Age did not emerge as a factor influencing CTR, i.e., there were no obvious trend and significant differences across age groups.

User Perception Besides user demographics, we examine the impact that diverse perceived factors can have on users based on explicit feedback collected from pre-task and post-task questionnaires.

User expected difficulty. Users engage more with CQs when completing a task that they expect to be more difficult (Figure 6.4). The one-way ANOVA test shows significant differences among different groups ($p<0.05$), and the post-hoc LSD test shows users completing search tasks of high difficulty level '4' are significantly more engaged with CQs than those engaging with tasks of lower levels of difficulty '1' , '2', and '3' ($p<0.05$). Moreover, when users expect a task to be more difficult, the answer CTR on C3 panes increases (46.3%, 64.2%, 67.2%, and 78.6% for difficulty level '1', '2', '3', and '4' respectively).

User perceived difficulty. Users engage more on C3 with the increase of user perceived difficulty level after completing the task (50.9%, 64.3%, 64.7%, 65.6% for difficulty level '1', '2', '3', and '4' respectively), which is the same trend with users' expected difficulty.

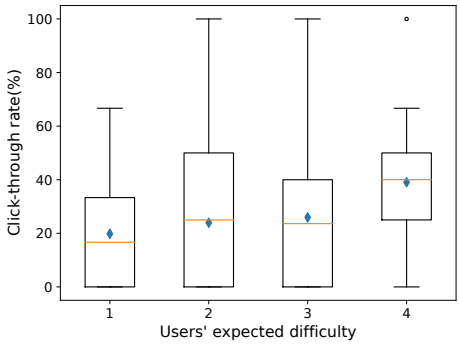


Figure 6.4: Expected task difficulty level on answer CTR.

User distraction. We expect users to be more eager to interact with CQs when they are distracted; indeed the overall answer CTR increases (not significantly) the more distracted users are (22.9%, 24.7%, 30.6% for 'not distracted', 'middle-level distracted', and 'highly distracted' respectively).

Search expertise. We posit that less-experienced users would be more willing to interact with CQs to successfully complete search tasks. Less experienced users (those using search engine weekly) indeed achieve higher mean and median values of overall answer CTR (24.2% vs. 31.3%) and answer CTR on C3 (61.7% vs. 66.7%) compared to those using search engines daily, however they also obtain a lower correct answer rate (55.1% vs. 33.3%). This indicates that adding a new feature to the search can be confusing to less experienced users.

We also considered users' prior knowledge on the task, users' task interest, perceived task clarity, perceived task relevance, and domain knowledge. We did not see any obvious trends or significant differences i.e., they seldom impact CTR.

6.4.3 **RQ5.3** User Engagement of CQs under Various Circumstances

We explore what circumstances lead to high engagement with CQs. Similar to [13, 139], we use two engagement metrics to measure user interest for CQs: (a) CQ answer CTR and (b) cursor hovering over the CQ pane.

CQ quality categories. We first examine how different quality categories of CQs affect answer CTR. As expected, CTR grows from C1 $\rightarrow$ C2 $\rightarrow$ C3. C3 achieves the highest CTR (61.9%) compared with C1 (10.6%), C2 (16.8%), and overall CTR (24.5%); significant differences between C1 and C2 (p<0.05), C1 and C3 (p<0.001), C2 and C3 (p<0.001). We attribute this to C3 being a related and useful CQ, i.e., the highest quality category in question quality compared to C1/C2. We also look at the impact of showing a bad quality CQ before a good quality CQ. Showing to users CQs of C1 or C2 before C3 CQs achieves lower CTR on C3 (50% and 46.9% respectively) compared to just showing C3 CQs (74.4%). This suggests that showing bad quality CQs before good quality CQs will lower the user engagement with CQs. We also observe that users

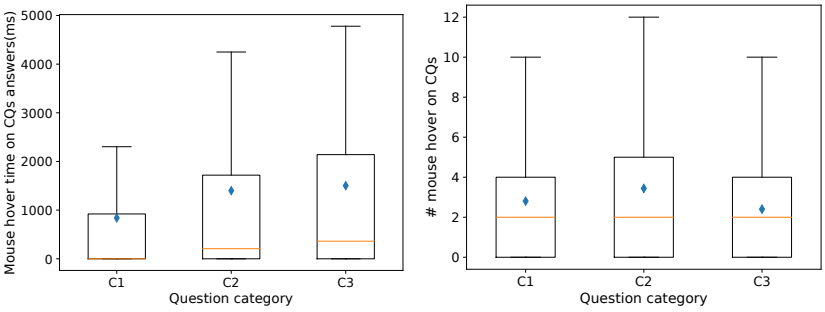


Figure 6.5: Cursor hovering on CQ quality categories.

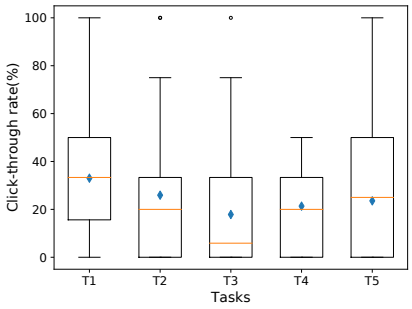


Figure 6.6: CTR by tasks.

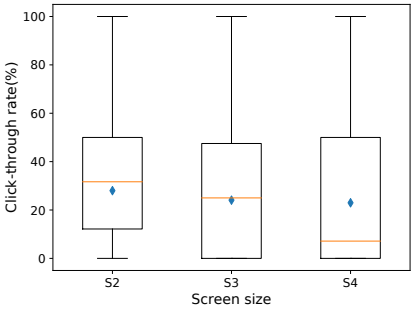


Figure 6.7: CTR by screen sizes.

still click C3 CQs even though they have seen the same CQs before but did not click on them, however with relatively low CTR (41.0%).

We investigate variations, if any, on cursor hover time and number of hovers across different quality CQs. As shown in Figure 6.5 (left), cursor hover time on answers of CQs per search request significantly increases from C1 to C2 (p<0.001) to C3 (p<0.001). In Figure 6.5 (right), we see that C2 obtains a significantly higher number of hovers on CQs pane (including questions and answers) over C3 (p<0.01) and C1 (p<0.05), which might be because of the confusion C2 creates to users with useful questions but useless answers.

Task types. Similar to previous studies indicating that search tasks may influence information-seeking behavior [75, 194], we see that the overall answer CTR also varies between tasks (Figure 6.6). The fact-finding tasks 'T1' and 'T2' have a higher overall CTR compared to the information gathering tasks 'T3', 'T4', and 'T5'. This might be because fact-finding tasks are simpler than information gathering ones [75] and users can foresee the benefit of answering the CQs. Also, we observe users rate higher expected difficulty for 'T1' and 'T2' in the questionnaire than 'T3', 'T4', and 'T5', which indicates that more difficult tasks attract more CTR. We also see that ambiguous tasks ('T3' and 'T4') received a lower overall CTR than faceted tasks ('T1', 'T2', and 'T5'), but with higher correct answer click rate ('T3': 0.66, 'T4': 0.63, 'T1': 0.45 'T2': 0.55, 'T5': 0.51). The ANOVA test shows significant differences among different tasks

(p<0.01). The post-hoc pairwise comparison shows significant differences between T1 and T3 (p<0.01), T1 and T4 (p<0.01), which are category-orthogonal tasks i.e., have no overlap for categories.

Query index. Zamani et al. [208] found that CTR increases for longer queries. Instead, we explore whether CTR increases with the growth of the query index (i.e., i-th query). We first compare user engagement between the first CQ pane shown to the user and subsequent ones, with the CTR being 39.9% and 19.3%, respectively. A similar picture is provided by the number of cursor hovers (mean: 3.57 vs. 2.73, p<0.01), and cursor hover time (mean: 1,885ms vs. 958ms, p<0.001). Moreover, among all CQs clicks, 41.4% occur the first time a CQ is shown to the user. This suggests that, for each search session, the first instance of showing a CQ is extremely important, and users pay attention to that. We also consider if the overall CTR increases as the query index does. We observe that users gradually lose their enthusiasm for the CQs as the query index increases (39.9%, 32.6%, 21.1%, 11.4%, 18.0%, 4.1%, 6.5%, 0%, 0% for CTR from 1-st query to 9-th query respectively).

SERP diversity and quality. To increase user satisfaction, search engines often show diverse results [196]. We explore whether more diverse SERPs prompt more CQ interactions. As in Web TREC, we use nERR-IA@10 as the metric of diversity. We divided nERR-IA@10 values into four equal bins: [0, 0.2), [0.2, 0.4), [0.4, 0.6), [0.6, 0.8). We observe that users engage more with CQs as SERP diversity increases (overall CTR: 22%, 22%, 29%, and 36% for [0, 0.2), [0.2, 0.4), [0.4, 0.6), and [0.6, 0.8) respectively).

As determined using NDCG@10 on CTR, when examining the impact of SERP quality we also see that users engage more with CQs with increasing SERP quality (overall CTR: 22%, 22%, 24%, and 36% for [0, 0.2), [0.2, 0.4), [0.4, 0.6), and [0.6, 0.8) respectively).

Screen size. According to the layout change of the CQ pane, we divided the screen size into 5 categories based on the screen width: 'S0' (width 0–690px), 'S1'(691–1063px), 'S2' (1064–1437px), 'S3' (1438–1811px), 'S4' (1812px–∞). From Figure 6.7, we see that a bigger screen gets a relatively lower overall CTR ('S2': 28.0%, 'S3': 24.1%, 'S4': 23.0%), but with higher correct answer click rate ('S2': 47.6%, 'S3': 57.3%, 'S4': 57.4%). Users with small screen size of S2 are engaging significantly more with CQs than with a large screen size of S4 (p<0.05). This might be because it is less practical to scan SERPs on small screen.

6.5 Conclusions and Discussion

In this chapter, we have presented a user study we conducted to investigate user interactions with search clarification panes in a web search setting. Our goal was to understand how adding such an element would affect users' experience in different ways and answer our research questions. Below, we summarize and discuss our findings.

RQ5.1*: Impact of CQs on search behavior and satisfaction.* Our analysis showed that user behavior and satisfaction are highly affected by the CQ quality. The query-level analysis indicates that interactions and user performance (relevant information found),

are significantly improved when the user is offered and click on a high quality CQ, whereas if CQs are of low or mid quality it is better for the user not to engage with them. This echoes the claims that users are willing to answer CQs if they are relevant and well-selected [226], suggesting that even though posing CQs to users causes higher interaction costs (in terms of time spent), it does not guarantee better returns. Therefore, future search systems should model the risk of asking low quality CQs and optimize their performance based on that [191].

Also, we observed that interacting with CQs affects user behavior and satisfaction in the entire session. Sessions including high quality CQs lead to higher session-based search performance compared to the ones without CQs. This is true even in the case that users did not actively engage with high quality CQs by clicking on their answers.

Moreover, we observed that while engaging with low or mid quality CQs decreases performance measures at the query-level, it still improves them at session-level. This suggests that despite the fact that these CQs lead to a worse immediate performance they may still help improve search performance for the session. A possible explanation is that low or mid quality CQs still offer implicit help to users by providing hints about the domain and the topic which could potentially aid user effectively reformulate their queries. In fact, one of our participants mentioned that "I forgot the name of a state so the question helped me clarify my search."

Last, clicking on high quality CQs improves user satisfaction, while users feel less satisfied when they skip CQs. This suggests that the interactivity has positive effects on user's perceived satisfaction and overall attitude [173], while showing CQs also introduces a risk of user dissatisfaction and frustration [56].

Our results can be used to inform the design of effective search clarification systems based on the query-level and session-level interactions of the users with different types of search topics and CQs. Even though this chapter focused on search systems, some of the findings can also be extended to conversational systems. For instance, in a conversational search setup, query difficulty prediction [8] could inform the system in choosing between asking a CQ or showing the answer to the user.

RQ5.2*: Impact of user perception on CQs interactions.* Our results showed that demographic traits and perception affect the way users interact with CQs. We observe that the user interactions with CQs are different for males and females, for participants with different language proficiency and educational backgrounds. Further, our results show that users engage significantly more with CQs when completing a task with higher expected difficulty. Other factors such as user perceived difficulty, distraction levels, and search expertise still affect user interaction with CQs but to a smaller degree. As indicated in Kim [99] there is a positive relationship between pre-task difficulty and web searching interactions like page viewing; similarly, our study suggests that the users' expected difficulty is an effective indicator of CQs interactions. Our observation for search expertise is inline with Kiesel et al. [97]: expertise using voice assistants has a weak effect.

Moreover, we see that users do not always click the correct answers, they also click wrong answers in all quality categories which is in line with the findings of Zou et al. [226] that users provide noisy answers for CQs and research should drop the assumption that all questions are answered correctly.

***RQ5.3**: User engagement with CQs under various circumstances.* User engagement with CQs is different for different CQ quality categories, task types, query index, degrees of SERP diversity and quality, and screen sizes. Specifically, we saw a much higher CTR on high quality CQs. More importantly, we observed that engagement decreases towards the end of a session. Also, we observed a bigger need for CQs on smaller screens (e.g., smartphones). Previous studies have found that small screen devices like smartphones are usually used on-the-go, leading to fragmented user attention [4, 78]. However, this phenomenon requires further investigation, under various interaction modalities (e.g., smartphone screen vs. voice [97]) and contexts (e.g., walking vs. driving [174]).

Moreover, we show that there are significant differences between the different quality categories of CQs for cursor hovering signals. This suggests that implicit user feedback like cursor hover time on answers of CQs, and the number of cursor hovers on the CQs pane could be effective indicators for CQ quality. As we observed significant differences in the way users interact with CQs of different quality (e.g., different cursor hover behavior), a combination of such signals can potentially be used to predict the quality of a CQ [164]. Predicted CQ quality can also be leveraged for the design of methods that learn from users' interactions [2, 16], as well as online evaluation methods [80, 101]. Moreover, they could be used as weak labels to train models to generate high quality CQ panes, especially in the case of commercial systems with a large number of interactions from multiple users against CQ panes.

This chapter depends on an online CQ-based system developed on the Clueweb09 corpus. Even though the user interface mimics the Bing's user interface, the retrieval effectiveness of Chatnoir is not on par with the effectiveness of Bing. This constitutes a limitation of our study given that search quality affects user engagement with CQ panes. A second limitation of this chapter is that the number of search tasks considered – only 5 – is limited. Many studies in our community (e.g., [34, 56, 78, 95, 96]) use a limited number of search tasks to allow control for multiple factors while keeping the cost of the study reasonable. Nevertheless, it would be beneficial to extend the analysis to more tasks in the future. Moreover, although users can answer multiple CQs in one search session, the choice of CQs shown to the user does not depend on the users' current query, since they are constructed prior to the search and are chosen randomly. Therefore, certain interactive and feedback effects are missing in this step. We leave the investigation of user behavior and question engagement in a multi-turn setting for the future. Another clear future direction is to consider different search settings, including different interfaces for interaction. For instance, it would be interesting to examine whether similar conclusions hold for voice-only conversational systems [4, 78].

In this chapter, to answer **RQ5**, we have conducted a large user study to understand user interactions with CQs for search clarification in web search. In contrast to Chapter 5 for exploring user willingness and the extent of providing correct answers to CQs in question-based product search, in this chapter, we analyzed the influence of user search performance, behavior and satisfaction by user interaction with CQs, as well as the need for user engagement with CQs in web search. Next, we conclude the thesis by providing the main findings and formulating ideas for future work.

7 Conclusions

In this chapter we conclude the thesis by summarizing our main findings in terms of our research questions from Chapter 1, and point out future directions.

7.1 Main Findings

RQ1 How can we ask clarifying questions (CQs) to effectively retrieve documents?

To answer this research question, we proposed a novel CQ-based approach SBSTAR for document search and validate its effectiveness. The method constructs a set of questions in terms of entities contained in the documents of the collection and directly asks yes/no questions to users about the expected presence of an entity in relevant documents so as to locate them. The model applies Generalized Binary Search (GBS) over entities to find the optimal sequence of CQs to ask and is updated based on Bayesian updating. To account for noisy user answers, we also provided a noise tolerant version of our algorithm. In addition, we also proposed an extension to decide when to stop asking questions to free the predefined parameter – the number of questions. The method is applied in the domain of systematic reviews, where total recall is essential. Typically, Continuous Active Learning (CAL) methods are applied but fail to find the last few relevant document. At that point our system starts asking CQs.

The experimental results on both abstract-level relevance and document-level relevance confirmed the effectiveness of our CQ-based document search method SBSTAR. SBSTAR can significantly reduce human effort, while achieving high recall, compared to state-of-the-art baselines. Also, we demonstrated the effectiveness of our algorithm under different settings, i.e., the noise tolerant version of our algorithm can achieve high performance and the extension that considers when to stop asking questions, learns to stop asking questions at the right time. Moreover, we discussed the impact of the stopping point, and the number of questions asked in our model. Last, besides the offline evaluation, the assumptions made regarding the users' willingness to answer a number of questions, their efforts, and their noisy answers were validated by a small online user study.

RQ2 How can we effectively ask CQs to improve product search?

To answer this research question, we proposed a novel CQ-based product search method, QSBPS, which directly queries users about the expected presence of an informative term in product related documents. The algorithm uses duet training, which learns question reward and cross-user system belief with limited data. The system belief training over products learns the interest of users over products, while entity effectiveness training over entities learns the reward or informativeness of questions, and thus finds the optimal policy for asking questions.

The experimental results confirmed the effectiveness of the proposed CQ-based product search method QSBPS. QSBPS can greatly improve the performance of product search compared to the state-of-the-art baselines. Also, we analyzed the effectiveness of different model components. We demonstrated that using user review data is helpful for the model performance, that using our duet learning framework is highly beneficial, and that the model can achieve good performance despite noise in answers. Moreover, we discussed the effectiveness of the QSBPS algorithm under different parameter settings, including the number of questions asked, and the question reward trade-off parameter.

RQ3 How can we effectively ask CQs to improve recommender system performance?

To answer this research question, we proposed a novel CQ-based recommendation model, Qrec, that combines the advantages of collaborative filtering based on matrix factorization and content analysis by querying users about descriptive item characteristics. The model firstly initializes the user and item latent factors offline and then updates the user and item latent factors online by incorporating feedback from the user based on our proposed matrix factorization algorithm, and also renews the user belief to select the next question to ask.

The evaluation results confirmed the effectiveness of the CQ-based recommendation method Qrec. The Qrec model achieved the highest performance compared to state-of-the-art baselines. We conducted a cold start experiment and showed that Qrec is effective in both user and item cold-start recommendation scenarios. We explored the contribution of the offline initialization module of Qrec and demonstrated that the offline initialization module is highly beneficial, especially at the early stage of question asking. Moreover, we discussed the effectiveness of Qrec under different parameter settings, including the online updating trade-off parameter, the dimension of the latent factors, and the number of questions asked. Finally, besides the offline evaluation, a small online user study was conducted to validate the the effectiveness of our method.

RQ4 To what extent can users answer CQs of question-based product search systems?

To answer this research question, we conducted an online user study for CQ-based product search system, in the domain of online retail. We explored the user willingness and user ability to answer CQs. We collected both interaction data and explicit feedback from users showing that: (a) users are willing to answer a good number of CQs (11 on average), but not many more than that; (b) most users answer questions until they reach the target product, but also a fraction of them stops due to fatigue or due to receiving irrelevant questions; (c) part of the users' answers (17%) are actually opposite to the description of the target product; while (d) most of the users (84%) find the question-based product search system helpful towards completing their product search

tasks. This suggests that related research should drop the assumption that users are happy to answer as many questions as the system generates and that all questions are answered correctly.

RQ5 How do users interact with CQs in web search?

To answer this research question, we conducted a large user study to understand user interaction with CQs for search clarification in a web search setting. We analyzed the impact of CQ interactions on user search behavior and satisfaction and the results show that when users engage with high quality CQ panes, the interaction, performance and satisfaction increases, compared to a search engine that does not offer such an option; however when the CQs are of low or mid quality they actually negatively affect all measures, even if they are presented to the user and the user does not engage with them. We also analyzed the impact of user background and user perception on user interactions, and we showed that users expected and perceived difficulty of the search task influenced the degree of their engagement with CQ, while less experienced users made a wrong use of the CQs, clicking more on irrelevant answers. Moreover, we explored the user engagement of CQs under various circumstances, and demonstrated that users' degree and quality of engagement with CQ panes are affected by factors such as SERP quality, diversity, screen size and decrease along with a search session.

7.2 Limitations and Future Directions

We first discuss the limitations and future directions for each research question answered by each corresponding chapter, and then the limitations and future directions regarding CQs for the entire thesis.

7.2.1 Question-based Document Search

In Chapter 2, we simulate user answers, noisy or not. In the no-noise setting, we assume that when presented with an entity reviewers know whether the entity is present in all missing documents with 100% confidence. In the noisy setting, we propose noise models to explore the performance of our algorithm SBSTAR and relax the aforementioned 100% confidence assumption. We define three different noise settings in Chapter 2. However, these three settings are ad-hoc and the noise can be also defined as any function of any characteristic of entities, users or topics. Our simulation setup is just a first step towards considering noisy answers, something that is utterly missing from past work on the topic. Our small user study indicates that there is validity in the assumptions we have made, but yet there is a need for an in-situ larger study to confirm the noise simulation.

Our when-to-stop algorithm $\text{SBSTAR}_{\text{ext}}$ is based on the extracted dynamic features. In Chapter 2 we extract seven related features to decide when to stop asking effectively. However, a systematic feature engineering investigation may discover more strong features related to automatically stop asking and feature selection algorithm may yield improvements, which we leave it as future work.

In Chapter 2 we interact with users by asking questions to locate the last few relevant documents. Another possible way of locating these last few relevant documents is to keep reformulating the query. Query reformulation has shown its effectiveness to locate the targets for initial query mismatching and limited coverage [48]. However, users need to find the association between queries and incorporate the new information gained from the previous search by themselves to reformulate the next query. Furthermore, query reformulation may generate some duplicate results and reviewing them will cost extra effort. Our work automatically selects questions to ask and incorporates the answers to refine the search results, which can be a complement of keeping query reformulation. One can also combine our method with query expansion or reformulation techniques to a guided query expansion or reformulation.

Last, in Chapter 2 we ask questions about informative terms to locate the last few relevant documents after deploying the CAL algorithm which queries on documents. Instead of asking questions after deploying the CAL algorithm, one can also explore the switch mechanism between asking questions and querying on document-based feedback in each iteration. For example, one can directly ask feedback on a document when the system is aware of enough confidence about it, while asking questions about faceted characteristics when there is low confidence about any certain documents.

7.2.2 Question-based Product Search

From a technical perspective, our work in Chapter 3 proposes a stand-alone algorithm that learns the informativeness of questions, along with user preferences. In principle, however, one can use a ranking method (any of the baselines mentioned in Chapter 3) to construct an informative prior belief on user preferences and reduce the number of necessary questions to find the product to smaller than 5. Further, one can also incorporate other factors (e.g., the importance level of different informative terms) to the objective function of question selection to extend the work.

Furthermore, in Chapter 3 we made the assumption that we know the topic of a user's query, so that we can load the right prior over preferences, and entity rewards. In practice, one needs some technique (of text similarity) to soft-match an arbitrary query to the already known, which we intent to explore in the future.

7.2.3 Question-based Recommendation

In Chapter 4, we uses a stand-alone algorithm that learns the informativeness of questions to ask based on GBS. One can also use other techniques (e.g., reinforcement learning) to learn the optimal question asking strategy, or incorporate more factors, e.g., the relatedness and importance level of different informative terms, to extend the work.

Also, the user may change their target item during the interaction with the system [140]. Theoretically the question-based recommendation method Qrec proposed in Chapter 4 is able to deal with this kind of situation, with new answers received gradually for the new target item. But we leave this experiment as future work.

7.2.4 A User Study on Question-based Product Search

One limitation of Chapter 5 is the isolated clarifying-based environment of the study. A more realistic experiment would require CQs to be embedded in an existing environment, where the user is enabled to not only answer questions, but also reformulate her query or filter results by selecting pre-defined item attributes, and browse the results to the preferred depth. Also a mixed-initiative approach under which a system switches from asking questions, to understanding user searches, and combining the two is worth studying. A further limitation of Chapter 5 is the fact that this was not an in-situ experiment by involving real users but a simulation of a use case of a question-based product search system by involving crowd workers. Hence, the findings are as good as our simulation of a user looking for a target product. Other factors, such as question quality, question format, and noisy answers, may affect the results, and studying therefore these factors in an A/B testing experiment would be beneficial. We leave all these as future work.

7.2.5 A User Study on Question-based Web Search

The study we conducted in Chapter 6 depends on an online CQ-based system developed on the Clueweb09 corpus. Even though the user interface mimics the Bing's user interface, the retrieval effectiveness of Chatnoir is not on par with the effectiveness of Bing. This constitutes a limitation of our study given that search quality affects user engagement with CQ panes.

A second limitation of this study is that the number of search tasks considered – only 5 – is limited. Many studies in our community (e.g., [34, 56, 78, 95, 96]) use a limited number of search tasks to allow control for multiple factors while keeping the cost of the study reasonable. Nevertheless, it would be beneficial to extend the analysis to more tasks in the future.

Moreover, although users can answer multiple CQs in one search session, the choice of CQs shown to the user does not depend on the users' current query, since they are constructed prior to the search and are chosen randomly. Therefore certain interactive and feedback effects are missing in this step. We leave the investigation of user behavior and question engagement in a multi-turn setting for the future.

Another clear future direction is to consider different search settings, including different interfaces for interaction. For instance, it would be interesting to examine whether similar conclusions hold for voice-only conversational systems [4, 78].

7.2.6 Clarifying Questions

In this thesis we pivot around the presence or absence of entities in the target documents to generate questions to ask to the user. We use the same rudimentary method of generating questions throughout the entire thesis. Recognizing entities in text is a research direction of its own, with significant recent work on neural methods, which further progresses the state of the art [76, 189]. In this thesis, we use TAGME, which is widely used in prior research, in semantic mapping [81, 198, 199], and biomedical information labeling [57, 137, 224]. However, this automatic entity annotation may

provide some irrelevant annotations or may miss some entities in the data. Our methods could certainly benefit from better entity recognition and salience detection methods, constructing more reliable and salient question pools. Similarly, there may be a richer set of possible questions to be asked, questions that may or may not be answered with a "yes" or a "no". For instance, questions could be constructed by using labeled topics [221, 223], keywords extraction [23], item categories and attributes, or other information extraction techniques [126, 149]. A richer type of questions could also be constructed by identifying properties of the documents (entities in a knowledge base triplet representation) and their relation to the document. For example, the following entities could be identified in the document description: "author", "year", "publisher", "subject category", patient population, intervention, comparison, outcome [187]. Questions then could be constructed from the derived triplets [148]. It is also likely that an entity may be semantically related to the desired document, while not lexically present. In this thesis we do not explore any semantic correlation modeling, but we leave it as future work.

In this thesis, we focus on asking CQs to assist search and recommender systems. However, it is also worth to further explore natural language conversations beyond CQs in future work [64]. Also, the CQs in this thesis are designed for a goal-directed purpose, one can incorporate chitchat conversations and goal-directed conversations to make the system's conversations more engaging and interactive [169].

In this thesis we do not learn an independent representation of CQs. Training a representation learning model for CQs by using deep learning techniques (e.g., transformer) to assist search and recommendation tasks is worth exploring [17]. For example, one can use the learned representation of CQs to expand the initial user query to enhance the performance of search systems.

A lot of research has been done on respecting ethical values and preserving privacy in information retrieval (IR) applications. When collecting user data by asking CQs, the ethical and privacy violations may occur. A typical solution is to anonymize the data and try to hide the identity of users [25], and encourage users to be careful about providing answers for ethical-sensitive and privacy-sensitive CQs. Also, a privacy preserving model can be trained when adding noise intentionally to the supervision signal to protect user privacy [54].

Bibliography

[1] M. Abualsaud, N. Ghelani, H. Zhang, M. D. Smucker, G. V. Cormack, and M. R. Grossman. A system for efficient high-recall retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1317–1320. 2018. (Cited on page 16.)

[2] A. Abujabal, R. S. Roy, M. Yahya, and G. Weikum. Never-ending learning for open-domain question answering over knowledge bases. In *WWW*, pages 1053–1062. 2018. (Cited on page 120.)

[3] Q. Ai, Y. Zhang, K. Bi, X. Chen, and W. B. Croft. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 645–654. 2017. (Cited on pages 47, 49, and 56.)

[4] M. Aliannejadi, M. Harvey, L. Costa, M. Pointon, and F. Crestani. Understanding mobile search task relevance and user behaviour in context. In *CHIIR*, pages 143–151. 2019. (Cited on pages 120 and 125.)

[5] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 475–484, 2019. (Cited on pages 2, 72, 101, 103, and 104.)

[6] M. Aliannejadi, J. Kiseleva, A. Chuklin, J. Dalton, and M. Burtsev. ConvAI3: Generating clarifying questions for open-domain dialogue systems (ClariQ). *arXiv preprint arXiv:2009.11352*, 2020. (Cited on pages 2, 101, and 104.)

[7] P. Ambrose and G. Johnson. A trust based model of buying behavior in electronic retailing. *AMCIS 1998 Proceedings*, page 91, 1998. (Cited on page 47.)

[8] N. Arabzadeh, F. Zarrinkalam, J. Jovanovic, and E. Bagheri. Neural embedding-based metrics for pre-retrieval query performance prediction. In *European Conference on Information Retrieval*, pages 78–85. 2020. (Cited on page 119.)

[9] A. Arampatzis, J. Kamps, and S. Robertson. Where to stop reading a ranked list? threshold optimization using truncated score distributions. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '09, pages 524–531. 2009. (Cited on page 19.)

[10] L. Azzopardi. Modelling interaction with economic models of search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 3–12, 2014. (Cited on page 112.)

[11] L. Azzopardi, M. Girolami, and K. Van Risjbergen. Investigating the relationship between language model perplexity and ir precision-recall measures. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 369–370, 2003. (Cited on page 109.)

[12] K. Bae and Y. Ko. Improving question retrieval in community question answering service using dependency relations and question classification. *Journal of the Association for Information Science and Technology*, 70(11):1194–1209, 2019. (Cited on pages 19 and 20.)

[13] J. Baird, N. Redmond, G. Harrison, B. Gebala, and J. Kawamoto. Systems and methods for capturing and reporting metrics regarding user engagement including a canvas model, July 12 2016. US Patent 9,390,438. (Cited on page 116.)

[14] J. R. Baron, D. D. Lewis, and D. W. Oard. TREC 2006 legal track overview. In *TREC*. 2006. (Cited on page 16.)

[15] N. J. Belkin, C. Cool, A. Stein, and U. Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications*, 9(3):379–395, 1995. (Cited on pages 2, 101, and 103.)

[16] M. Bendersky, X. Wang, D. Metzler, and M. Najork. Learning from user interactions in personal search via attribute parameterization. In *WSDM*, pages 791–799. 2017. (Cited on page 120.)

[17] K. Bi, Q. Ai, Y. Zhang, and W. B. Croft. Conversational product search based on negative feedback. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 359–368. 2019. (Cited on page 126.)

[18] J. Bobadilla, F. Ortega, A. Hernando, and A. Gutiérrez. Recommender systems survey. *Knowledge-based systems*, 46:109–132, 2013. (Cited on page 1.)

[19] P. Borlund. The IIR evaluation model: a framework for evaluation of interactive information retrieval systems. *Information research*, 8(3):8–3, 2003. (Cited on page 106.)

[20] P. Braslavski, D. Savenkov, E. Agichtein, and A. Dubatovka. What do you mean exactly? Analyzing

clarification questions in CQA. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, pages 345–348, 2017. (Cited on page 104.)

[21] D. G. Bridge. Towards conversational recommender systems: A dialogue grammar approach. In *ECCBR Workshops*, pages 9–22, 2002. (Cited on page 71.)

[22] C. Buckley and S. Robertson. Relevance feedback track overview: TREC 2008. In *TREC '08*, 2008. (Cited on page 19.)

[23] R. Campos, V. Mangaravite, A. Pasquali, A. M. Jorge, C. Nunes, and A. Jatowt. A text feature based automatic keyword extraction method for single documents. In *Advances in Information Retrieval*, pages 684–691. 2018. (Cited on pages 22, 45, 66, and 126.)

[24] G. Carenini, J. Smith, and D. Poole. Towards more conversational and collaborative recommender systems. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 12–18. 2003. (Cited on page 71.)

[25] C. Carpineto and G. Romano. Semantic search log k-anonymization with generalized k-cores of query concept graph. In *European Conference on Information Retrieval*, pages 110–121. 2013. (Cited on page 126.)

[26] J. Y. Chai, C. Zhang, and R. Jin. An empirical investigation of user term feedback in text-based targeted image search. *ACM Transactions on Information Systems (TOIS)*, 25(1):3, 2007. (Cited on page 19.)

[27] Q. Chen, J. Lin, Y. Zhang, M. Ding, Y. Cen, H. Yang, and J. Tang. Towards knowledge-based recommender dialog system. *arXiv preprint arXiv:1908.05391*, 2019. (Cited on pages 1, 72, and 91.)

[28] Y. Chen, B. Chen, X. Duan, J.-G. Lou, Y. Wang, W. Zhu, and Y. Cao. Learning-to-ask: Knowledge acquisition via 20 questions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, pages 1216–1225. 2018. (Cited on pages 50 and 72.)

[29] Y. Chen, Y. Wang, X. Zhao, J. Zou, and M. de Rijke. Block-aware item similarity models for top-n recommendation. *ACM Trans. Inf. Syst.*, 38(4), Sept. 2020.

[30] Z. Chen, C. Zhang, Z. Zhao, C. Yao, and D. Cai. Question retrieval for community-based question answering via heterogeneous social influential network. *Neurocomputing*, 285:117 – 124, 2018. (Cited on pages 19 and 20.)

[31] K. Christakopoulou, F. Radlinski, and K. Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 815–824. 2016. (Cited on pages 1, 69, 70, 71, 74, and 79.)

[32] K. Christakopoulou, A. Beutel, R. Li, S. Jain, and E. H. Chi. Q&R: A two-stage approach toward interactive recommendation. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 139–148. 2018. (Cited on page 72.)

[33] A. Coden, D. Gruhl, N. Lewis, and P. N. Mendes. Did you mean a or b? Supporting clarification dialog for entity disambiguation. In *SumPre-HSWI@ ESWC*, 2015. (Cited on page 104.)

[34] K. Collins-Thompson, S. Y. Rieh, C. C. Haynes, and R. Syed. Assessing learning outcomes in web search: A comparison of tasks and query strategies. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 163–172, 2016. (Cited on pages 120 and 125.)

[35] G. V. Cormack and M. R. Grossman. Evaluation of machine-learning protocols for technology-assisted review in electronic discovery. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 153–162. 2014. (Cited on pages 1, 13, 14, 16, 18, 19, 24, and 49.)

[36] G. V. Cormack and M. R. Grossman. Autonomy and reliability of continuous active learning for technology-assisted review. *arXiv preprint arXiv:1504.06868*, 2015. (Cited on pages 19 and 49.)

[37] G. V. Cormack and M. R. Grossman. Waterloo (Cormack) participation in the TREC 2015 total recall track. In *TREC*, 2015. (Cited on page 18.)

[38] G. V. Cormack and M. R. Grossman. Engineering quality and reliability in technology-assisted review. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 75–84. 2016. (Cited on page 18.)

[39] G. V. Cormack and M. R. Grossman. Scalability of continuous active learning for reliable high-recall text classification. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1039–1048. 2016. (Cited on page 18.)

[40] G. V. Cormack and M. R. Grossman. "When to stop" Waterloo (Cormack) participation in the TREC 2016 total recall track. In *TREC*, 2016. (Cited on page 18.)

[41] G. V. Cormack and M. R. Grossman. Technology-assisted review in empirical medicine: Waterloo participation in CLEF eHealth 2017. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017. (Cited on pages 14, 18, 28, 29, 32,

and 35.)
[42] G. V. Cormack and M. R. Grossman. Beyond pooling. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1169–1172. 2018. (Cited on page 13.)
[43] G. V. Cormack and T. R. Lynam. TREC 2005 spam track overview. In *TREC*, pages 500–274, 2005. (Cited on page 24.)
[44] G. V. Cormack and M. Mojdeh. Machine learning for information retrieval: TREC 2009 web, relevance feedback and legal tracks. In *TREC*, 2009. (Cited on pages 18 and 19.)
[45] G. V. Cormack, C. R. Palmer, and C. L. A. Clarke. Efficient construction of large test collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 282–289. 1998. (Cited on page 19.)
[46] G. V. Cormack, M. R. Grossman, B. Hedin, and D. W. Oard. Overview of the TREC 2010 legal track. In *Proc. 19th Text REtrieval Conference*, volume 1, 2010. (Cited on page 16.)
[47] M. Cornolti, P. Ferragina, M. Ciaramita, S. Rüd, and H. Schütze. Smaph: A piggyback approach for entity-linking in web queries. *ACM Transactions on Information Systems (TOIS)*, 37(1):13, 2018. (Cited on page 22.)
[48] V. Dang and B. W. Croft. Query reformulation using anchor text. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining*, WSDM '10, pages 41–50. 2010. (Cited on pages 46 and 124.)
[49] M. De Boni and S. Manandhar. An analysis of clarification dialogue for question answering. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–55, 2003. (Cited on pages 2 and 103.)
[50] G. M. Di Nunzio. A study of an automatic stopping strategy for technologically assisted medical reviews. In *Advances in Information Retrieval*, pages 672–677. 2018. (Cited on pages 18, 28, and 29.)
[51] H. Drucker, B. Shahrary, and D. Gibbon. Relevance feedback using support vector machines. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 122–129. 2001. (Cited on page 24.)
[52] H. Duan, C. Zhai, J. Cheng, and A. Gattani. A probabilistic mixture model for mining and analyzing product search log. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, CIKM '13, pages 2179–2188. 2013. (Cited on pages 47 and 48.)
[53] H. Duan, C. Zhai, J. Cheng, and A. Gattani. Supporting keyword search in product database: A probabilistic approach. *Proc. VLDB Endow.*, 6(14):1786–1797, Sept. 2013. (Cited on pages 47 and 48.)
[54] J. C. Duchi, M. I. Jordan, and M. J. Wainwright. Privacy aware learning. *Journal of the ACM (JACM)*, 61(6):1–57, 2014. (Cited on page 126.)
[55] M. Dudík, K. Hofmann, R. E. Schapire, A. Slivkins, and M. Zoghi. Contextual dueling bandits. *arXiv preprint arXiv:1502.06362*, 2015. (Cited on page 19.)
[56] A. Edwards and D. Kelly. Engaged or frustrated?: Disambiguating emotional state in search. In *SIGIR*, pages 125–134. 2017. (Cited on pages 101, 119, 120, and 125.)
[57] P. Ernst, A. Mishra, A. Anand, and V. Setty. Bionex: A system for biomedical news event exploration. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1277–1280. 2017. (Cited on pages 22, 45, and 125.)
[58] E. Erosheva, S. Fienberg, and J. Lafferty. Mixed-membership models of scientific publications. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5220–5227, 2004. (Cited on page 22.)
[59] A. Felfernig, G. Friedrich, D. Jannach, and M. Zanker. Developing constraint-based recommenders. In *Recommender systems handbook*, pages 187–215. Springer, 2011. (Cited on page 71.)
[60] G. Ferguson, J. F. Allen, B. W. Miller, et al. Trains-95: Towards a mixed-initiative planning assistant. In *AIPS*, pages 70–77, 1996. (Cited on pages 2 and 101.)
[61] P. Ferragina and U. Scaiella. Tagme: On-the-fly annotation of short text fragments (by wikipedia entities). In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1625–1628. 2010. (Cited on pages 14, 22, 51, and 77.)
[62] N. Ferro and C. Peters. *Information Retrieval Evaluation in a Changing World: Lessons Learned from 20 Years of CLEF*, volume 41. Springer, 2019. (Cited on page 16.)
[63] Z. Fu, H. Gao, W. Guo, S. K. Jha, J. Jia, X. Liu, B. Long, J. Shi, S. Wang, and M. Zhou. Deep learning for search and recommender systems in practice. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 3515–3516, 2020. (Cited on page 1.)
[64] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua. Advances and challenges in conversational

recommender systems: A survey. *arXiv preprint arXiv:2101.09459*, 2021. (Cited on page 126.)

[65] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010. (Cited on page 40.)

[66] L. Goeuriot, L. Kelly, H. Suominen, A. Névéol, A. Robert, E. Kanoulas, R. Spijker, J. Palotti, and G. Zuccon. CLEF 2017 eHealth evaluation lab overview. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 291–303. 2017. (Cited on page 28.)

[67] M. P. Graus and M. C. Willemsen. Improving the user experience during cold start through choice-based preference elicitation. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 273–276. 2015. (Cited on page 71.)

[68] C. Greco, A. Suglia, P. Basile, and G. Semeraro. Converse-et-impera: Exploiting deep learning and hierarchical reinforcement learning for conversational recommender systems. In *Conference of the Italian Association for Artificial Intelligence*, pages 372–386. 2017. (Cited on page 72.)

[69] M. R. Grossman and G. V. Cormack. Technology-assisted review in e-discovery can be more effective and more efficient than exhaustive manual review. *Rich. JL & Tech.*, 17:1, 2010. (Cited on page 19.)

[70] M. R. Grossman, G. V. Cormack, and A. Roegiest. TREC 2016 total recall track overview. In *TREC '16*, 2016. (Cited on pages 14 and 16.)

[71] M. R. Grossman, G. V. Cormack, and A. Roegiest. Automatic and semi-automatic document selection for technology-assisted review. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 905–908. 2017. (Cited on pages 1, 14, and 18.)

[72] Y. Guo, Z. Cheng, L. Nie, X.-S. Xu, and M. Kankanhalli. Multi-modal preference modeling for product search. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, pages 1865–1873. 2018. (Cited on pages 47, 49, and 57.)

[73] Y. Guo, Z. Cheng, L. Nie, Y. Wang, J. Ma, and M. Kankanhalli. Attentive long short-term preference modeling for personalized product search. *ACM Transactions on Information Systems (TOIS)*, 37(2): 19, 2019. (Cited on pages 47, 49, and 58.)

[74] T. Gupta. Keyword extraction: a review. *International Journal of Engineering Applied Sciences and Technology*, 2(4):215–220, 2017. (Cited on page 51.)

[75] J. Gwizdka. Revisiting search task difficulty: Behavioral and individual difference measures. *Proceedings of the American Society for Information Science and Technology*, 45(1):1–12, 2008. (Cited on pages 107 and 117.)

[76] K. Hakala and S. Pyysalo. Biomedical named entity recognition with multilingual BERT. In *Proceedings of The 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61. Nov. 2019. (Cited on pages 45 and 125.)

[77] L. Han, E. Maddalena, A. Checco, C. Sarasua, U. Gadiraju, K. Roitero, and G. Demartini. Crowd worker strategies in relevance judgment tasks. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 241–249, 2020. (Cited on page 109.)

[78] M. Harvey and M. Pointon. Searching on the go: The effects of fragmented attention on mobile web search tasks. In *SIGIR*, pages 155–164. 2017. (Cited on pages 120 and 125.)

[79] H. Hashemi, H. Zamani, and W. B. Croft. Guided transformer: Leveraging multiple external sources for representation learning in conversational search. In *SIGIR*, pages 1131–1140. 2020. (Cited on pages 2, 101, 103, and 104.)

[80] S. H. Hashemi, K. Williams, A. E. Kholy, I. Zitouni, and P. A. Crook. Measuring user satisfaction on smart speaker intelligent assistants using intent sensitive query embeddings. In *CIKM*, pages 1183–1192. 2018. (Cited on page 120.)

[81] F. Hasibi, K. Balog, and S. E. Bratsberg. Entity linking in queries: Tasks and evaluation. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 171–180. 2015. (Cited on pages 22, 45, and 125.)

[82] C. He, D. Parra, and K. Verbert. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Systems with Applications*, 56:9–27, 2016. (Cited on page 71.)

[83] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. 2017. (Cited on pages 70, 71, and 79.)

[84] B. Hedin, S. Tomlinson, J. R. Baron, and D. W. Oard. Overview of the TREC 2009 legal track. In *The Eighteenth Text REtrieval Conference (TREC 2009)*, 2009. (Cited on page 16.)

[85] K. Hofmann, S. Whiteson, and M. de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

(Cited on pages 19 and 49.)
[86] W. Hong, S. Zheng, and H. Wang. Dynamic user profile-based job recommender system. In *2013 8th International Conference on Computer Science & Education*, pages 1499–1503. 2013. (Cited on page 1.)
[87] H. Hu, X. Wu, B. Luo, C. Tao, C. Xu, W. Wu, and Z. Chen. Playing 20 question game with policy-based reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3233–3242. Oct.-Nov. 2018. (Cited on pages 50 and 72.)
[88] R. Hu and P. Pu. Acceptance issues of personality-based recommender systems. In *Proceedings of the third ACM conference on Recommender systems*, pages 221–224, 2009. (Cited on page 1.)
[89] J. Huang, R. W. White, G. Buscher, and K. Wang. Improving searcher models using mouse cursor activity. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 195–204, 2012. (Cited on page 110.)
[90] J. Jiang and J. Allan. Correlation between system and user metrics in a session. In *Proceedings of the 2016 ACM on Conference on Human Information Interaction and Retrieval*, pages 285–288, 2016. (Cited on page 114.)
[91] Y. Jin, W. Cai, L. Chen, N. N. Htun, and K. Verbert. Musicbot: Evaluating critiquing-based music recommenders with conversational interaction. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 951–960, 2019. (Cited on page 71.)
[92] M. Jugovac and D. Jannach. Interacting with recommenders – overview and research directions. *ACM Trans. Interact. Intell. Syst.*, 7(3):10:1–10:46, Sept. 2017. (Cited on page 71.)
[93] E. Kanoulas, D. Li, L. Azzopardi, and R. Spijker. CLEF 2017 technologically assisted reviews in empirical medicine overview. In *Working Notes of CLEF 2017 - Conference and Labs of the Evaluation Forum, Dublin, Ireland, September 11-14, 2017.*, 2017. (Cited on pages 14, 16, 18, 19, 23, 28, 29, and 30.)
[94] J. Kawale, H. H. Bui, B. Kveton, L. Tran-Thanh, and S. Chawla. Efficient Thompson sampling for online matrix-factorization recommendation. In *Advances in neural information processing systems*, pages 1297–1305, 2015. (Cited on pages 71 and 74.)
[95] D. Kelly and L. Azzopardi. How many results per page? A study of SERP size, search behavior and user experience. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 183–192, 2015. (Cited on pages 109, 120, and 125.)
[96] D. Kelly, J. Arguello, A. Edwards, and W. Wu. Development and evaluation of search tasks for IIR experiments using a cognitive complexity framework. In *ICTIR*, pages 101–110. 2015. (Cited on pages 120 and 125.)
[97] J. Kiesel, A. Bahrami, B. Stein, A. Anand, and M. Hagen. Toward voice query clarification. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1257–1260, 2018. (Cited on pages 102, 104, 115, 119, and 120.)
[98] D.-Y. Kim, X. Y. Lehto, and A. M. Morrison. Gender differences in online travel information search: Implications for marketing communications on the internet. *Tourism management*, 28(2):423–433, 2007. (Cited on page 115.)
[99] J. Kim. Task difficulty as a predictor and indicator of web searching interaction. In *CHI'06 extended abstracts on human factors in computing systems*, pages 959–964, 2006. (Cited on page 119.)
[100] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. (Cited on page 75.)
[101] J. Kiseleva, K. Williams, J. Jiang, A. H. Awadallah, A. C. Crook, I. Zitouni, and T. Anastasakos. Understanding user satisfaction with intelligent assistants. In *CHIIR*, pages 121–130. 2016. (Cited on page 120.)
[102] N. Kondylidis, J. Zou, and E. Kanoulas. Category aware explainable conversational recommendation. In *Workshop on Mixed-Initiative ConveRsatiOnal Systems 2021*, 2021.
[103] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009. (Cited on page 74.)
[104] A. M. Krasakis, M. Aliannejadi, N. Voskarides, and E. Kanoulas. Analysing the effect of clarifying questions on document ranking in conversational search. In *Proceedings of the 2020 ACM SIGIR on International Conference on Theory of Information Retrieval*, pages 129–132, 2020. (Cited on pages 2, 101, and 104.)
[105] A. Krishnakumar. Active learning literature survey. Technical report, Technical Report, University of California, Santa Cruz, 2007. (Cited on page 19.)
[106] D. Kundu and D. P. Mandal. Formulation of a hybrid expertise retrieval system in community question answering services. *Applied Intelligence*, 49(2):463–477, Feb. 2019. (Cited on page 20.)

[107] B. Kveton and S. Berkovsky. Minimal interaction search in recommender systems. In *Proceedings of the 20th International Conference on Intelligent User Interfaces*, IUI '15, pages 236–246. 2015. (Cited on page 20.)

[108] B. Kveton and S. Berkovsky. Minimal interaction content discovery in recommender systems. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(2):15, 2016. (Cited on page 20.)

[109] V. Lavrenko and W. B. Croft. Relevance-based language models. In *ACM SIGIR Forum*, volume 51, pages 260–267. 2017. (Cited on page 19.)

[110] G. E. Lee and A. Sun. Seed-driven document ranking for systematic reviews in evidence-based medicine. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 455–464. 2018. (Cited on pages 28 and 29.)

[111] W. Lei, X. He, Y. Miao, Q. Wu, R. Hong, M.-Y. Kan, and T.-S. Chua. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, WSDM '20, page 304–312. 2020. (Cited on page 72.)

[112] B. Li and I. King. Routing questions to appropriate answerers in community question answering services. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management*, CIKM '10, pages 1585–1588. 2010. (Cited on page 20.)

[113] R. Li, S. E. Kahou, H. Schulz, V. Michalski, L. Charlin, and C. Pal. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems*, pages 9748–9758, 2018. (Cited on pages 1, 69, 70, 72, and 91.)

[114] L. Liao, R. Takanobu, Y. Ma, X. Yang, M. Huang, and T.-S. Chua. Deep conversational recommender in travel. *arXiv preprint arXiv:1907.00710*, 2019. (Cited on page 72.)

[115] S. C. J. Lim, Y. Liu, and W. B. Lee. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing & Management*, 46(4):479 – 493, 2010. (Cited on page 48.)

[116] D. Lin and P. Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001. (Cited on page 20.)

[117] M. Liu, L. Chen, B. Liu, G. Zheng, and X. Zhang. DBpedia-based entity linking via greedy search and adjusted monte carlo random walk. *ACM Transactions on Information Systems (TOIS)*, 36(2):16, 2017. (Cited on page 22.)

[118] M. Liu, G. Gong, B. Qin, and T. Liu. A multi-view–based collective entity linking method. *ACM Transactions on Information Systems (TOIS)*, 37(2):23, 2019. (Cited on page 22.)

[119] M. Liu, J. Mao, Y. Liu, M. Zhang, and S. Ma. Investigating cognitive effects in session-level search user satisfaction. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 923–931, 2019. (Cited on page 114.)

[120] Y. Liu, S. Li, Y. Cao, C.-Y. Lin, D. Han, and Y. Yu. Understanding and summarizing answers in community-based question answering services. In *Proceedings of the 22nd International Conference on Computational Linguistics - Volume 1*, COLING '08, pages 497–504. 2008. (Cited on pages 19 and 20.)

[121] B. Loepp, T. Hussein, and J. Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3085–3094. 2014. (Cited on page 71.)

[122] D. E. Losada, J. Parapar, and A. Barreiro. When to stop making relevance judgments? a study of stopping methods for building information retrieval test collections. *Journal of the Association for Information Science and Technology*, 70(1):49–60, 2019. (Cited on page 19.)

[123] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 255–264. 2009. (Cited on pages 19 and 49.)

[124] T. Mahmood and F. Ricci. Learning and adaptivity in interactive recommender systems. In *Proceedings of the Ninth International Conference on Electronic Commerce*, ICEC '07, pages 75–84. 2007. (Cited on page 71.)

[125] T. Mahmood and F. Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. 2009. (Cited on page 71.)

[126] M. Maslennikov and T.-S. Chua. Combining relations for information extraction from free text. *ACM Transactions on Information Systems (TOIS)*, 28(3):14, 2010. (Cited on pages 22, 45, and 126.)

[127] J. McAuley, R. Pandey, and J. Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery*

and Data Mining, KDD '15, pages 785–794. 2015. (Cited on page 56.)
[128] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52. 2015. (Cited on page 79.)
[129] G. McDonald, C. Macdonald, and I. Ounis. Active learning strategies for technology assisted sensitivity review. In *Advances in Information Retrieval*, pages 439–453. 2018. (Cited on page 18.)
[130] A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008. (Cited on pages 71 and 74.)
[131] R. Nowak. Generalized binary search. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 568–574. Sep. 2008. (Cited on pages 48 and 70.)
[132] D. W. Oard, B. Hedin, S. Tomlinson, and J. R. Baron. Overview of the TREC 2008 legal track. In *TREC '08*, 2008. (Cited on page 16.)
[133] D. W. Oard, F. Sebastiani, and J. K. Vinjumur. Jointly minimizing the expected costs of review for responsiveness and privilege in e-discovery. *ACM Transactions on Information Systems (TOIS)*, 37(1): 11, 2018. (Cited on page 13.)
[134] H. L. O'Brien, J. Arguello, and R. Capra. An empirical study of interest, task complexity, and search behaviour on user engagement. *Information Processing & Management*, 57(3):102226, 2020. (Cited on page 92.)
[135] A. O'Mara-Eves, J. Thomas, J. McNaught, M. Miwa, and S. Ananiadou. Using text mining for study identification in systematic reviews: a systematic review of current approaches. *Systematic Reviews*, 4 (1):5, Jan 2015. (Cited on pages 2, 13, and 14.)
[136] K. D. Onal, Y. Zhang, I. S. Altingovde, M. M. Rahman, P. Karagoz, A. Braylan, B. Dang, H.-L. Chang, H. Kim, Q. McNamara, et al. Neural information retrieval: At the end of the early years. *Information Retrieval Journal*, 21(2):111–182, 2018. (Cited on page 1.)
[137] M. Park, H. Sampathkumar, B. Luo, and X.-w. Chen. Content-based assessment of the credibility of online healthcare information. In *2013 IEEE International Conference on Big Data*, pages 51–58. 2013. (Cited on pages 22, 45, and 125.)
[138] M. J. Pazzani and D. Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007. (Cited on page 71.)
[139] T. Piccardi, M. Redi, G. Colavizza, and R. West. Quantifying engagement with citations on wikipedia. In *Proceedings of The Web Conference 2020*, pages 2365–2376, 2020. (Cited on page 116.)
[140] C. Plua and A. Jameson. Collaborative preference elicitation in a group travel recommender system. In *Proceedings of the AH 2002 Workshop on Recommendation and Personalization in eCommerce*, 2002. (Cited on pages 87 and 124.)
[141] M. Potthast, M. Hagen, B. Stein, J. Graßegger, M. Michel, M. Tippmann, and C. Welsch. ChatNoir: A search engine for the clueweb09 corpus. In *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval*, pages 1004–1004, 2012. (Cited on page 106.)
[142] B. Priyogi. Preference elicitation strategy for conversational recommender system. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 824–825. 2019. (Cited on page 69.)
[143] P. Qi, Y. Zhang, and C. D. Manning. Stay hungry, stay focused: Generating informative and specific questions in information-seeking conversations. *arXiv preprint arXiv:2004.14530*, 2020. (Cited on pages 1 and 91.)
[144] F. Radlinski and N. Craswell. A theoretical framework for conversational search. In *Proceedings of the 2017 conference on conference human information interaction and retrieval*, pages 117–126, 2017. (Cited on pages 2, 101, 103, and 104.)
[145] F. Radlinski, R. Kleinberg, and T. Joachims. Learning diverse rankings with multi-armed bandits. In *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pages 784–791. 2008. (Cited on page 19.)
[146] F. Radlinski, K. Balog, B. Byrne, and K. Krishnamoorthi. Coached conversational preference elicitation: A case study in understanding movie preferences. In *SIGdial*, pages 353–360. 2019. (Cited on pages 2 and 101.)
[147] H. Raviv, O. Kurland, and D. Carmel. Document retrieval using entity-based language models. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 65–74. 2016. (Cited on pages 22 and 51.)
[148] S. Reddy, D. Raghu, M. M. Khapra, and S. Joshi. Generating natural language question-answer pairs from a knowledge graph using a RNN based question generation model. In *Proceedings of the 15th*

Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 376–385. Apr. 2017. (Cited on pages 45, 51, 66, and 126.)

[149] E. Riloff and W. Lehnert. Information extraction as a basis for high-precision text classification. *ACM Transactions on Information Systems (TOIS)*, 12(3):296–333, 1994. (Cited on pages 22, 45, and 126.)

[150] S. E. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, 27(3):129–146, 1976. (Cited on page 19.)

[151] J. J. Rocchio. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing*, pages 313–323, 1971. (Cited on page 19.)

[152] A. Roegiest, G. V. Cormack, M. R. Grossman, and C. Clarke. TREC 2015 total recall track overview. *Proc. TREC-2015*, 2015. (Cited on pages 16 and 24.)

[153] M. Rosen-Zvi, T. Griffiths, M. Steyvers, and P. Smyth. The author-topic model for authors and documents. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, UAI '04, pages 487–494. 2004. (Cited on page 22.)

[154] C. Rosset, C. Xiong, X. Song, D. Campos, N. Craswell, S. Tiwary, and P. Bennett. Leading conversational search by suggesting useful questions. In *Proceedings of The Web Conference 2020*, WWW '20, page 1160–1170. 2020. (Cited on pages 103 and 107.)

[155] J. Rowley. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing*, 17(1):20–35, 2000. (Cited on pages 47 and 48.)

[156] T. Ruotsalo, J. Peltonen, M. J. Eugster, D. Głowacka, P. Floréen, P. Myllymäki, G. Jacucci, and S. Kaski. Interactive intent modeling for exploratory search. *ACM Transactions on Information Systems (TOIS)*, 36(4):44, 2018. (Cited on page 19.)

[157] G. Salton and C. Buckley. Improving retrieval performance by relevance feedback. *Journal of the American society for information science*, 41(4):288–297, 1990. (Cited on page 19.)

[158] M. Sanderson. Accurate user directed summarization from existing tools. In *Proceedings of the Seventh International Conference on Information and Knowledge Management*, CIKM '98, pages 45–51. 1998. (Cited on pages 18 and 44.)

[159] M. Sanderson and H. Joho. Forming test collections with no system pooling. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '04, pages 33–40. 2004. (Cited on page 13.)

[160] N. Sardella, C. Biancalana, A. Micarelli, and G. Sansonetti. An approach to conversational recommendation of restaurants. In *International Conference on Human-Computer Interaction*, pages 123–130. 2019. (Cited on page 71.)

[161] V. Satopaa, J. Albrecht, D. Irwin, and B. Raghavan. Finding a "kneedle" in a haystack: Detecting knee points in system behavior. In *2011 31st International Conference on Distributed Computing Systems Workshops*, pages 166–171, June 2011. (Cited on page 18.)

[162] H. Scells, L. Azzopardi, G. Zuccon, and B. Koopman. Query variation performance prediction for systematic reviews. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1089–1092. 2018. (Cited on pages 28 and 29.)

[163] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen. *Collaborative Filtering Recommender Systems*, pages 291–324. Springer Berlin Heidelberg, 2007. (Cited on pages 69 and 70.)

[164] I. Sekulic, M. Aliannejadi, and F. Crestani. User engagement prediction for clarification in search. In *Proceedings of the European Conference on Informatin Retrieval (ECIR)*, 2021. (Cited on page 120.)

[165] A. Sepliarskaia, J. Kiseleva, F. Radlinski, and M. de Rijke. Preference elicitation as an optimization problem. In *RecSys*, pages 172–180. 2018. (Cited on pages 101 and 103.)

[166] I. Soboroff and S. Robertson. Building a filtering test collection for TREC 2002. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval*, SIGIR '03, pages 243–250. 2003. (Cited on page 18.)

[167] K. Spark-Jones. Report on the need for and provision of an 'ideal' information retrieval test collection. *Computer Laboratory*, 1975. (Cited on pages 18 and 19.)

[168] S. Stoyanchev, A. Liu, and J. Hirschberg. Towards natural clarification questions in dialogue systems. In *AISB symposium on questions, discourse and dialogue*, volume 20, 2014. (Cited on pages 2 and 103.)

[169] K. Sun, S. Moon, P. Crook, S. Roller, B. Silvert, B. Liu, Z. Wang, H. Liu, E. Cho, and C. Cardie. Adding chit-chats to enhance task-oriented dialogues. *arXiv preprint arXiv:2010.12757*, 2020. (Cited on page 126.)

[170] Y. Sun and Y. Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 235–244. 2018. (Cited on pages 1, 56, 69, 70, 72, 79, 80, and 91.)

[171] H. Suominen, L. Kelly, L. Goeuriot, A. Névéol, L. Ramadier, A. Robert, E. Kanoulas, R. Spijker, L. Azzopardi, D. Li, Jimmy, J. Palotti, and G. Zuccon. Overview of the CLEF eHealth evaluation lab 2018. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction*, pages 286–301. 2018. (Cited on pages 14 and 16.)

[172] J. Teevan, S. T. Dumais, and E. Horvitz. Personalizing search via automated analysis of interests and activities. In *SIGIR*, pages 449–456. 2005. (Cited on page 101.)

[173] H.-H. Teo, L.-B. Oh, C. Liu, and K.-K. Wei. An empirical study of the effects of interactivity on web user attitude. *International journal of human-computer studies*, 58(3):281–305, 2003. (Cited on page 119.)

[174] M. ter Hoeve, R. Sim, E. Nouri, A. Fourney, M. de Rijke, and R. W. White. Conversations with documents: An exploration of document-centered assistance. In *CHIIR*, pages 43–52. 2020. (Cited on page 120.)

[175] C. A. Thompson, M. H. Goker, and P. Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004. (Cited on page 71.)

[176] A. Tombros and M. Sanderson. Advantages of query biased summaries in information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 2–10. 1998. (Cited on pages 18 and 101.)

[177] S. Tomlinson, D. W. Oard, J. R. Baron, and P. Thompson. Overview of the TREC 2007 legal track. In *TREC*. 2007. (Cited on page 16.)

[178] J. R. Trippas, D. Spina, L. Cavedon, and M. Sanderson. How do people interact in conversational speech-only search tasks: A preliminary analysis. In *Proceedings of the 2017 Conference on Conference Human Information Interaction and Retrieval*, CHIIR '17, pages 325–328. 2017. (Cited on page 104.)

[179] D. Vallet and P. Castells. Personalized diversification of search results. In *SIGIR*, pages 841–850. 2012. (Cited on page 101.)

[180] C. Van Gysel, M. de Rijke, and E. Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 165–174. 2016. (Cited on pages 47, 48, 56, 57, and 79.)

[181] D. Vandic, J.-W. van Dam, and F. Frasincar. Faceted product search powered by the semantic web. *Decision Support Systems*, 53(3):425 – 437, 2012. (Cited on page 48.)

[182] D. Vandic, F. Frasincar, and U. Kaymak. Facet selection algorithms for web product search. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2327–2332. 2013. (Cited on page 48.)

[183] E. M. Voorhees and D. K. Harman. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge, 2005. (Cited on page 16.)

[184] A. Vtyurina, D. Savenkov, E. Agichtein, and C. L. Clarke. Exploring conversational search with humans, assistants, and wizards. In *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, pages 2187–2193. 2017. (Cited on page 104.)

[185] B. C. Wallace, T. A. Trikalinos, J. Lau, C. Brodley, and C. H. Schmid. Semi-automated screening of biomedical citations for systematic reviews. *BMC bioinformatics*, 11(1):55, 2010. (Cited on page 18.)

[186] B. C. Wallace, I. J. Dahabreh, K. H. Moran, C. E. Brodley, and T. A. Trikalinos. Active literature discovery for scoping evidence reviews: How many needles are there. In *KDD workshop on data mining for healthcare (KDD-DMH)*, 2013. (Cited on page 19.)

[187] B. C. Wallace, J. Kuiper, A. Sharma, M. Zhu, and I. J. Marshall. Extracting PICO sentences from clinical trial reports using supervised distant supervision. *The Journal of Machine Learning Research*, 17(1):4572–4596, 2016. (Cited on pages 45 and 126.)

[188] Y. Wan, G. Xu, L. Chen, Z. Zhao, and J. Wu. Exploiting cross-source knowledge for warming up community question answering services. *Neurocomputing*, 320:25 – 34, 2018. (Cited on page 20.)

[189] X. Wang, Y. Zhang, X. Ren, Y. Zhang, M. Zitnik, J. Shang, C. Langlotz, and J. Han. Cross-type biomedical named entity recognition with deep multi-task learning. *Bioinformatics*, 35(10):1745–1752, 2019. (Cited on pages 45 and 125.)

[190] Y. Wang, C. Liu, M. Huang, and L. Nie. Learning to ask questions in open-domain conversational systems with typed decoders. *arXiv preprint arXiv:1805.04843*, 2018. (Cited on page 72.)

[191] Z. Wang and Q. Ai. Controlling the risk of conversational search via reinforcement learning, 2021. (Cited on pages 2, 101, 103, 107, and 119.)

[192] I. Weber and A. Jaimes. Who uses web search for what: And how. In *Proceedings of the fourth ACM international conference on Web search and data mining*, pages 15–24, 2011. (Cited on page 114.)

[193] Z. Wen, B. Kveton, B. Eriksson, and S. Bhamidipati. Sequential bayesian search. In *International*

Conference on Machine Learning, pages 226–234. 2013. (Cited on pages 14, 20, 49, 50, 51, 53, 54, 57, 76, 77, and 80.)

[194] R. W. White, M. Bilenko, and S. Cucerzan. Studying the use of popular destinations to enhance web search interaction. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 159–166, 2007. (Cited on pages 106 and 117.)

[195] G. Wu, K. Luo, S. Sanner, and H. Soh. Deep language-based critiquing for recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 137–145, 2019. (Cited on page 71.)

[196] Z. Wu, K. Zhou, Y. Liu, M. Zhang, and S. Ma. Does diversity affect user satisfaction in image search. *ACM Transactions on Information Systems (TOIS)*, 37(3):1–30, 2019. (Cited on page 118.)

[197] B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *SIGIR*, pages 451–458. 2010. (Cited on page 101.)

[198] C. Xiong and J. Callan. Esdrank: Connecting query and documents through external semi-structured data. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 951–960. 2015. (Cited on pages 22, 45, 51, and 125.)

[199] C. Xiong, J. Callan, and T.-Y. Liu. Word-entity duet representations for document ranking. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 763–772. 2017. (Cited on pages 22, 45, 51, and 125.)

[200] J. Xu, X. He, and H. Li. Deep learning for matching in search and recommendation. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1365–1368, 2018. (Cited on pages 1 and 3.)

[201] J. Xu, Y. Wang, D. Tang, N. Duan, P. Yang, Q. Zeng, M. Zhou, and S. Xu. Asking clarification questions in knowledge-based question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1618–1629, 2019. (Cited on page 103.)

[202] T. Yu, Y. Shen, and H. Jin. An visual dialog augmented interactive recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 157–165. 2019. (Cited on page 71.)

[203] Z. Yu, N. A. Kraft, and T. Menzies. How to read less: Better machine assisted reading methods for systematic literature reviews. *arXiv preprint arXiv:1612.03224*, 2016. (Cited on page 16.)

[204] Z. Yu, N. A. Kraft, and T. Menzies. Finding better active learners for faster literature reviews. *Empirical Software Engineering*, 23(6):3161–3186, 2018. (Cited on page 18.)

[205] H. Zamani and W. B. Croft. Joint modeling and optimization of search and recommendation. *arXiv preprint arXiv:1807.05631*, 2018. (Cited on page 49.)

[206] H. Zamani, S. Dumais, N. Craswell, P. Bennett, and G. Lueck. Generating clarifying questions for information retrieval. In *Proceedings of The Web Conference 2020*, pages 418–428, 2020. (Cited on pages 2, 101, 102, 103, 104, 106, and 107.)

[207] H. Zamani, G. Lueck, E. Chen, R. Quispe, F. Luu, and N. Craswell. Mimics: A large-scale data collection for search clarification. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 3189–3196, 2020. (Cited on pages 2, 101, 103, and 104.)

[208] H. Zamani, B. Mitra, E. Chen, G. Lueck, F. Diaz, P. N. Bennett, N. Craswell, and S. T. Dumais. Analyzing and learning from user interactions for search clarification. *arXiv preprint arXiv:2006.00166*, 2020. (Cited on pages 102, 104, 106, 107, and 118.)

[209] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 403–410. 2001. (Cited on page 19.)

[210] H. Zhang, W. Lin, Y. Wang, C. L. Clarke, and M. D. Smucker. Waterlooclarke: TREC 2015 total recall track. In *TREC*, 2015. (Cited on page 16.)

[211] H. Zhang, M. Abualsaud, N. Ghelani, M. D. Smucker, G. V. Cormack, and M. R. Grossman. Effective user interaction for high-recall retrieval: Less is more. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 187–196. 2018. (Cited on page 18.)

[212] H. Zhang, G. V. Cormack, M. R. Grossman, and M. D. Smucker. Evaluating sentence-level relevance feedback for high-recall information retrieval. *arXiv preprint arXiv:1803.08988*, 2018. (Cited on pages 18 and 44.)

[213] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft. Towards conversational search and recom-

mendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 177–186. 2018. (Cited on pages 1, 2, 30, 31, 49, 56, 57, 69, 70, 72, 79, 80, 83, 91, 101, 103, and 104.)

[214] F. Zhao, M. Xiao, and Y. Guo. Predictive collaborative filtering with side information. In *IJCAI*, pages 2385–2391, 2016. (Cited on page 71.)

[215] X. Zhao, W. Zhang, and J. Wang. Interactive collaborative filtering. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 1411–1420. 2013. (Cited on pages 71 and 79.)

[216] Z. Zhao, L. Zhang, X. He, and W. Ng. Expert finding for question answering via graph regularized matrix completion. *IEEE Transactions on Knowledge and Data Engineering*, 27(4):993–1004, 2014. (Cited on pages 19 and 20.)

[217] Z. Zhao, Q. Yang, D. Cai, X. He, and Y. Zhuang. Expert finding for community-based question answering via ranking metric network learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 3000–3006. 2016. (Cited on pages 19 and 20.)

[218] K. Zhou, W. X. Zhao, H. Wang, S. Wang, F. Zhang, Z. Wang, and J.-R. Wen. Leveraging historical interaction data for improving conversational recommender system. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2349–2352, 2020. (Cited on page 104.)

[219] J. Zou and E. Kanoulas. Learning to ask: Question-based sequential bayesian product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, CIKM '19, pages 369–378, 2019. (Cited on pages 1, 2, 30, 47, 69, 70, 72, 77, 79, 80, 91, 93, 101, 103, and 104.)

[220] J. Zou and E. Kanoulas. Towards question-based high-recall information retrieval: Locating the last few relevant documents for technology-assisted reviews. *ACM Trans. Inf. Syst.*, 38(3), May 2020. (Cited on pages 13 and 80.)

[221] J. Zou, L. Xu, W. Guo, M. Yan, D. Yang, and X. Zhang. Which non-functional requirements do developers focus on? an empirical study on stack overflow using topic analysis. In *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, pages 446–449. 2015. (Cited on pages 22, 45, 87, and 126.)

[222] J. Zou, L. Xu, M. Yang, X. Zhang, J. Zeng, and S. Hirokawa. Automated duplicate bug report detection using multi-factor analysis. *IEICE Transactions on Information and Systems*, E99.D(7):1762–1775, 2016. (Cited on page 51.)

[223] J. Zou, L. Xu, M. Yang, X. Zhang, and D. Yang. Towards comprehending the non-functional requirements through developers' eyes: An exploration of stack overflow using topic analysis. *Information and Software Technology*, 84:19–32, 2017. (Cited on pages 22, 45, 51, 66, 87, and 126.)

[224] J. Zou, D. Li, and E. Kanoulas. Technology assisted reviews: Finding the last few relevant documents by asking yes/no questions to reviewers. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 949–952, 2018. (Cited on pages 1, 13, 14, 18, 19, 20, 21, 22, 23, 25, 44, 45, 50, 53, 54, 70, 76, 77, 80, 91, and 125.)

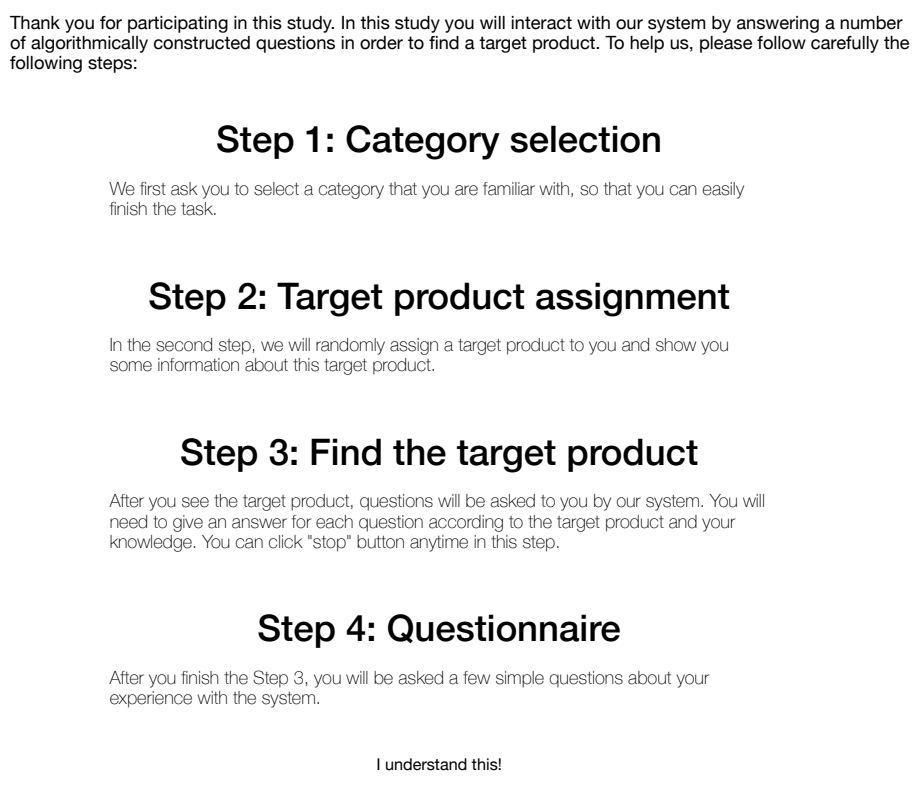
[225] J. Zou, Y. Chen, and E. Kanoulas. Towards question-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, pages 881–890, 2020. (Cited on pages 1, 2, 69, 91, 92, 101, 103, and 104.)

[226] J. Zou, E. Kanoulas, and Y. Liu. An empirical study on clarifying question-based systems. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2361–2364, 2020. (Cited on pages 91, 103, 104, 107, and 119.)

[227] J. Zou, M. Aliannejadi, E. Kanoulas, M. S. Pera, and Y. Liu. Users meet clarifying questions: Towards a better understanding of user interactions for search clarification. *ACM Trans. Inf. Syst.*, 2021. Submitted. (Cited on page 101.)

A Instructions for the User Study in Chapter 5

At the beginning of the user study in Chapter 5, we offer participants detailed instructions for the study. We guide the participants by going through 4 steps to complete the study, with a corresponding brief description of each step, as shown below:



B Instructions for the User Study in Chapter 6

At the beginning of the user study in Chapter 6, we offer participants detailed instructions for the study and clarify its goal. We outline the steps to complete the study, with a corresponding brief description of each step. We also provide visual examples, an introductory video, and some tips to help participants understand the study:

Thank you for agreeing to take part of this study, where you will interact with our search tool. To complete the required study, please follow the steps outlined below. Please use desktop computers or laptops to do this study and allow all cookies in your browser during this study. P.S. We value your privacy. As such, we will not report identifiable information on any of the manuscripts and other public materials that will result from this study.

Step 1: Demographic survey

We would like to know some general information about you, so we will first ask you to complete a survey related to demographic data. This will help us understand the effect of our system on different groups of people in the society.

Step 2: Topic selection

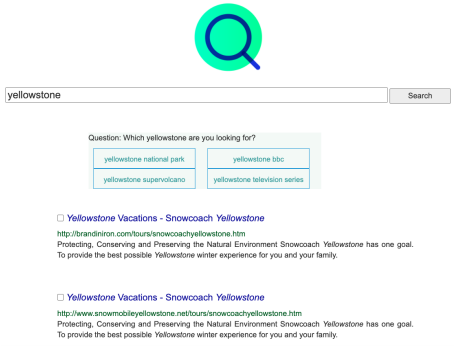
We will present you with a number of topics and then ask you to search for information related to said topics. Please read through the topic descriptions and select the one that you feel most comfortable about--this will help you more easily complete the search task that is the focus of this study.

Step 3: Pre-task Questionnaire

Once you have selected a topic, we will ask you to complete a pre-task questionnaire, so that we can gauge how familiar you are with your chosen topic.

Step 4: Find the information

We will now ask you to find some information about your topic of choice. This is the main part of the study. To achieve this goal, you will interact with our system, which is very similar to search engines like Google and Bing. You will see one difference: when you perform a search, you will probably also see "clarifying questions". As you complete the assigned search task, you can interact with our system as you would usually do with any other search tool, but you can also engage with the possibly presented clarifying questions. In the image below, you can see an example search page for the query "yellowstone":



(Continued)

As you see, after submitting a query you will see the corresponding generated results, together with the "clarifying question" (but note: the "clarifying question" not always appears with every search). Please:

- Think of a query based on the topic you selected, type the query in the search box and wait for search results together with a clarifying question (if any).
- Try reading the summaries of results and click on the ones that you find interesting and would like to know more about. A search result looks like below:

- You can reformulate query by yourself or click an answer on the clarifying question if you find it useful, to generate more results. (Note: not all presented clarifying questions are useful, some clarifying questions maybe unrelated or useless and you don't have to interact with every presented clarifying question. An automatical reformulated query will be generated by our system after you click an answer).
- You will also see a checkbox, right next to the result title. Use this checkbox to mark the results that you find useful and relevant to the search topic. Below you can see a sample checked result: ☑ *Yellowstone* Vacations - Snowcoach *Yellowstone*

To ensure you understand it, we also recorded a video for this step. Please click here to watch it (https://youtu.be/PLKmj6Imnmw) (NOTE: watching this video (~80s) is mandatory). After you watch it, please back to this page to finish the study.

Notes:

- You are free to interact with the search tool in any way you like, as long as you find the information you are looking for. During your search, you are allowed to:
  - Browse and scan the results,
  - Change your search query and repeat the search as many times as you wish,
  - Answer the clarifying questions.
- The search engine may be a bit slow, please be patient after you submit your query and wait for the results.

Step 5: Post-task Questionnaire

After you finish the search, you can click the "next step" button as shown in top right corner of the search page, then you will be asked a few simple questions about your experience interacting with our system. Specifically, we would like to understand your experience engaging with our system and the ease (or lack thereof) of using clarifying questions to complete search tasks.

I understand this!

Summary

Recently, there has been renewed interest in interactive search and recommender systems through the development of systems that ask clarifying questions (CQs) to users to better understand their information needs. Although the significance and effectiveness of CQs have been recongnized, research on modelling CQs to improve search and recommender systems is limited, compared to traditional search and recommender systems. Also, understanding users' willingness or ability to answer these CQs, and the extent to which CQs impact users' behavior and their ability to identify relevant information remains relatively unexplored.

The work in this thesis provides a series of models and online user studies for search and recommender systems that construct and use CQs. In the first part of the thesis, we focus on CQ-based search and recommendation algorithms. We first propose a novel CQ-based document search algorithm, Sequential Bayesian Search based method for Technology-Assisted Review (SBSTAR) and its extension, to efficiently retrieve the last few, but significant, relevant documents. The algorithm sequentially selects and directly asks CQs to users about the presence or absence of an entity in the missing relevant documents. Next, we propose a novel CQ-based product search algorithm, Question-based Sequential Bayesian Product Search (QSBPS), to effectively locate products that users are looking for. The model is based on a duet learning framework that learns product relevance and the reward of the potential CQ to be asked to the user. Furthermore, we propose a novel CQ-based recommendation algorithm, Question-based recommendation (Qrec), to assist users to find items interactively. The model is first trained offline by a novel matrix factorization algorithm, and then iteratively updates the user and item latent factors online based on the user's answers. We experimentally test the performance of the proposed models and demonstrate that they outperform state-of-the-art baselines.

In the second part of the thesis, we focus on the evaluation of CQ-based search and recommendation. We first conduct an online user study by deploying a CQ-based system in the domain of online retail, to understand to what extent users can answer CQs. We explore the user willingness, ability, and user perception to provide answers to CQs. We find that users are willing to answer a good number of the system generated questions and most users answer questions until they reach the target product. Users also provide incorrect answers while answering CQs, and most users are positive towards CQ-based systems. Next, we conduct a large user study on web search to understand the impact of CQ interactions on user search behavior and satisfaction, and how users interact with different levels of quality of CQs under different user perceptions and conditions. We find that user interactions with high quality CQs improve user performance and satisfaction, while low and mid quality CQs are harmful. We also observe that user engagement, and therefore the user need for CQ support, is affected by various factors, such as search result quality or perceived task difficulty.

Samenvatting

Onlangs is er hernieuwde belangstelling voor interactieve zoek- en aanbevelingssystemen door de ontwikkeling van systemen die verduidelijkende vragen (VV's) stellen aan gebruikers om hun informatiebehoeften beter te begrijpen. Hoewel het belang en de effectiviteit van VV's zijn onderkend, is onderzoek naar het modelleren van VV's om zoek- en aanbevelingssystemen te verbeteren beperkt in vergelijking met traditionele zoek- en aanbevelingssystemen. Daarnaast is het begrijpen van de bereidheid of het vermogen van gebruikers om deze VV's te beantwoorden, en de mate waarin VV's van invloed zijn op het gedrag van gebruikers en hun vermogen om relevante informatie te identificeren, nog relatief onontgonnen terrein.

Het werk in dit proefschrift biedt een reeks modellen en online gebruikersstudies voor zoek- en aanbevelingssystemen die VV's construeren en gebruiken. In het eerste deel van het proefschrift richten we ons op VV-gebaseerde zoek- en aanbevelingsalgoritmen. We stellen eerst een nieuw, op VV's gebaseerd, zoekalgoritme voor documenten voor, Sequential Bayesian Search based method for Technology-Assisted Review (SB-STAR) en de uitbreiding ervan, om de laatste, maar belangrijke relevante documenten efficiënt op te halen. Het algoritme selecteert VV's over de aan- of afwezigheid van een entiteit in de ontbrekende relevante documenten en stelt ze vervolgens direct aan gebruikers. Vervolgens stellen we een nieuw VV-gebaseerd zoekalgoritme voor producten voor, Question-based Sequential Bayesian Product Search (QSBPS), om effectief producten te vinden waarnaar gebruikers op zoek zijn. Het model is gebaseerd op een duaal leerraamwerk dat zowel de productrelevantie leert als de beloning (reward) voor de potentiële VV die aan de gebruiker moet worden gesteld. Daarnaast stellen we ook een nieuw VV-gebaseerd aanbevelingsalgoritme voor, Question-based recommendation (Qrec), om gebruikers te helpen om items op interactieve wijze te vinden. Het model wordt eerst offline getraind door een nieuw matrix factorization algoritme, en werkt vervolgens iteratief en online de latente factoren van de gebruiker en het item bij, op basis van de antwoorden van de gebruiker. We testen de prestaties van de voorgestelde modellen door middel van experimenten en laten zien dat ze beter presteren dan de state-of-the-art baselines.

In het tweede deel van het proefschrift richten we ons op de evaluatie van VV-gebaseerd zoeken en aanbevelen. We voeren eerst een online gebruikersonderzoek uit door een VV-gebaseerd systeem in te zetten voor de online detailhandel, om zo te kunnen begrijpen in hoeverre gebruikers VV's kunnen beantwoorden. We onderzoeken de bereidheid, het vermogen en de gebruikersperceptie van gebruikers om antwoorden op VV's te geven. We ontdekken dat gebruikers bereid zijn om een groot aantal door het systeem gegenereerde vragen te beantwoorden en dat de meeste gebruikers vragen beantwoorden totdat ze het doelproduct hebben bereikt. Gebruikers geven ook foute antwoorden tijdens het beantwoorden van VV's en de meeste gebruikers staan positief tegenover VV-gebaseerde systemen. Vervolgens voeren we een groot gebruikersonderzoek uit naar het zoeken op internet om de impact van VV-interacties op het zoekgedrag en de tevredenheid van gebruikers te begrijpen, en om te begrijpen hoe gebruikers omgaan met verschillende kwaliteitsniveaus van VV's onder verschillende gebruikerspercepties en omstandigheden. We vinden dat gebruikersinteracties met VV's van hoge kwaliteit de prestaties en tevredenheid van gebruikers verbeteren, terwijl

VV's van lage en gemiddelde kwaliteit schadelijk zijn. We zien ook dat gebruikersbetrokkenheid, en dus de gebruikersbehoefte aan VV-ondersteuning, wordt beïnvloed door verschillende factoren, zoals de kwaliteit van het zoekresultaat of de gepercipieerde moeilijkheidsgraad van de taak.