UNIVERSITY OF AMSTERDAM

UvA-DARE (Digital Academic Repository)

Query auto completion in information retrieval

Cai, Fei

Publication date
2016
Document Version
Final published version

Link to publication

Citation for published version (APA):
Cai, F. (2016). *Query auto completion in information retrieval*. [Thesis, fully internal, Universiteit van Amsterdam].

General rights
It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

Disclaimer/Complaints regulations
If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: https://uba.uva.nl/en/contact, or a letter to: Library of the University of Amsterdam, Secretariat, Singel 425, 1012 WP Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (https://dare.uva.nl)

Download date:08 Mar 2023

Query auto completion is an important feature embedded into today's search engines. It can help users formulate queries which other people have searched for when he/she finishes typing the query prefix. Today's most sophisticated query auto completion approaches are based on the collected query logs to provide the best possible queries for each searcher's input.

In this thesis, we develop new query auto completion methods for information retrieval. First, we consider the information of both time and user to propose a time-sensitive personalized query auto completion approach. In previous work, these two sources of information have been developed separately. We bring them together and pay special attention to long-tail prefixes. Second, based on a learning-to-rank framework, we propose to extract features originating from so-called homologous queries and from the semantic similarity of terms, which allow the contributions from similar queries and from semantic relatedness to be used for query auto completion.

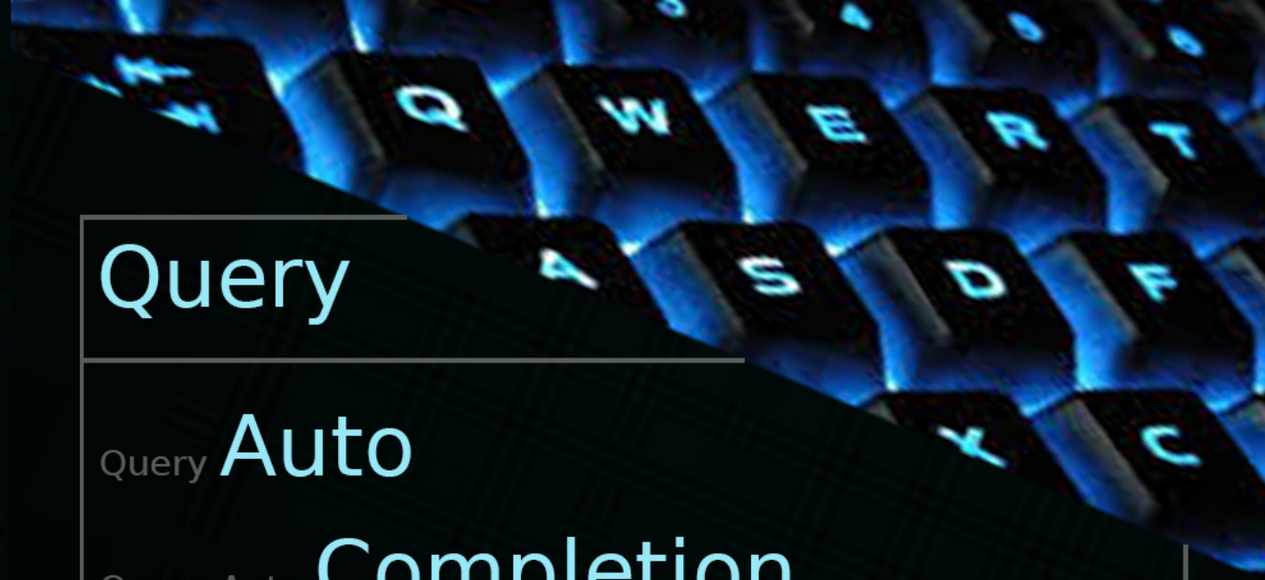
In addition, we study the problem of query auto completion diversification, where we aim to diversify aspect-level query intents of query completions. This task has not been studied before. Given that only a limited number of query completions can be returned to users of a search engine, it is important to remove redundant queries and improve user satisfaction by finding an acceptable query. Finally, we conduct an investigation on when to personalize query auto completion by proposing a selectively personalizing query auto completion approach, where the weight of personalization in a query auto completion model is selectively assigned based on the search context in session.

The experimental results in this thesis indicate that our proposed query auto completion approaches can improve the ranking performance of query completions in terms of well-known metrics, like Mean Reciprocal Rank. The unique insights and interesting findings in this thesis may be used to help search engine designers.

ISBN 978-94-6182-688-6

Query Auto Completion in Information Retrieval

Fei Cai



Query Auto Completion in Information Retrieval

Fei Cai

Query Auto Completion in Information Retrieval

Fei Cai

Query Auto Completion in Information Retrieval

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan de
Universiteit van Amsterdam
op gezag van de Rector Magnificus
prof.dr. D.C. van den Boom
ten overstaan van een door het college voor promoties ingestelde
commissie, in het openbaar te verdedigen in
de Agnietenkapel
op dinsdag 14 juni 2016, te 14:00 uur

door

Fei Cai

geboren te Jiangsu, China

Promotiecommissie

Promotor:		
	Prof. dr. M. de Rijke	Universiteit van Amsterdam
Co-promotor:		
	Prof. dr. H. Chen	National University of Defense Technology
Overige leden:		
	Dr. E. Kanoulas	Universiteit van Amsterdam
	Prof. dr. C. de Laat	Universiteit van Amsterdam
	Dr. C. Monz	Universiteit van Amsterdam
	Prof. dr. ir. A.P. de Vries	Radboud Universiteit
	Prof. S. Wu	Jiangsu University

Faculteit der Natuurwetenschappen, Wiskunde en Informatica

SIKS Dissertation Series No. 2016-23
The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

The research was supported by the China Scholarship Council.

Copyright © 2016 Fei Cai, Amsterdam, The Netherlands
Cover by Fei Cai
Printed by Off Page, Amsterdam

ISBN: 978-94-6182-688-6

Acknowledgements

Although only my name appears on the cover of this dissertation, many people have contributed to its production. I owe my sincere gratitude to all those people who have made this dissertation possible and who have made my research experience one that I will cherish forever.

My deepest gratitude is to my promotor Prof.dr. Maarten de Rijke for his continuous support of my Ph.D study and related research. His patience, motivation, support and immense knowledge have helped me throughout the research and writing of this dissertation. I could not have imagined having a better promotor for my Ph.D. studies. I am very proud of working with Maarten in the past two and half years for my Ph.D. degree on computer science in University of Amsterdam. I am also thankful to my co-promotor Prof.dr. Honghui Chen. He has always been supporting me to finish my Ph.D. at the University of Amsterdam. I am deeply grateful to him for his long-term encouragement and discussions that helped me sort out the details of my work.

I am very honored and grateful to have Evangelos Kanoulas, Cees de Laat, Christof Monz, Arjen de Vries, and Shengli Wu serving as my committee members.

I thank all the members of the Information and Language Processing Systems (ILPS) group at the University of Amsterdam. I am thankful to Auke and Isaac for setting up and maintaining the computing infrastructure for ILPS, helping me implement the experiments successfully. My thanks go to Aleksandr, Alexey, Anne, Arianna, Artem, Cristina, Christophe, Daan, Damien, David vD, David G, Hamid, Hosein, Ilya, Katya, Ke, Lars, Marlies, Marzieh, Masrour, Nikos, Praveen, Ridho, Richard, Tom, Xinyi, Zhaochun and others in the IAS and ISIS groups, Guangliang, Nihang, Ran, Zhenyang and Zhongyu, for discussing and enriching my ideas. I enjoyed working with you! I thank Caroline and Petra for helping me address countless practical details, allowing me to focus on my research only.

I would like to acknowledge the China Scholarship Council (CSC) for the financial support for two years that funded parts of the research discussed in this dissertation. I also appreciate the generosity of my home university, the National University of Defense Technology, for supporting my study abroad.

Finally, I would like to thank my family, especially my wife Junlin and my daughter Xinran, as well as many friends, for supporting me spiritually throughout writing this thesis and my life in general. Their support and care helped me overcome setbacks and stay focused on my study. I deeply appreciate their belief in me.

Fei Cai
Changsha
March 14, 2016

Contents

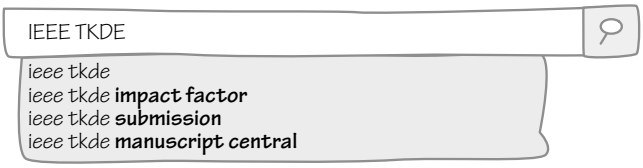
1 Introduction

Information Retrieval (IR) aims to address searchers' information needs. Common search activities often involve someone submitting a query to a search engine and receiving answers in the form of a list of documents in ranked order. The primary focus of the IR field since the 1950s has been on *text documents*, e.g., web pages, emails, scholarly papers, books, and news stories (Croft et al., 2015), which typically have a special structure to store the information like title, author, date, and abstract. The elements of the structure are often called attributes or fields. The important distinction between a document in IR and a typical database record, such as a bank account record or a flight reservation, is that most of the information in an IR-related document is in the form of text, which is relatively unstructured (Croft et al., 2015). A basic goal of the research in IR is to develop the insights and ideas needed to generate a ranked result list to respond queries from users, and thus to meet their information acquisition.

Before ranking documents, a search system should receive a *query* from their users. After that, the relevance of documents to this particular query can be estimated, by which the documents are then sorted. Query formulation thus was born to produce such queries to be consumed by the search engine, where typically a text corpus is involved for term weighting and query expansion related query formulation activities. The main process of query formulation refers to query suggestion, query rewriting and query transformation, etc., with the aim to better represent the underlying intent of the user. Thus, the primary goal of query formulation is to improve the overall quality of the document ranking presented to the user in response to their query. As a member of the family of query formulation tasks, Query Auto Completion (QAC) has as its task to help the user formulate her query while she is typing only the prefix, e.g., several characters (Bast and Weber, 2006). The main purpose of QAC is to predict the user's intended query and thereby save her keystrokes. In addition, with the advent of instant as-you-type search results (e.g., Google Instant[1]), correct query prediction has become very important, because it determines the speed at which the user sees the suitable results for her intended information request while being engaged in search (Bar-Yossef and Kraus, 2011). QAC has become a prominent features of today's major search engines, e.g., Bing, Google and Yahoo!, as well as some popular online properties, e.g., online shopping sites and email services. In pre-computed auto completion systems, the list of matching candidates for a prefix is generated in advance and stored in an efficient data structure for fast lookup.

[1] `http://www.google.com/instant/`

(a) Query auto completion of the prefix *"IEEE␣"*.



(b) Query auto completion of the prefix *"IEEE TKDE"*.

Figure 1.1: (Top) Query auto completion for the prefix *"IEEE␣"*, where ␣ indicates that a space follows after *"IEEE"*. (Bottom) The refined completions after continuing to type *TKDE* after *"IEEE␣"*.

When needed, as shown in Figure 1.1, continued typing characters can lead to dynamic refinements of the completions by exact prefix matching according to the user's input prefix until an appropriate completion is found. Where offered, the facility is heavily used and highly influential on search results (Bar-Yossef and Kraus, 2011; Shokouhi and Radinsky, 2012).

A straightforward and useful approach in previous work on QAC is to extract past queries with each prefix from a period of query logs, and then rank them by their popularity (Bar-Yossef and Kraus, 2011; Shokouhi and Radinsky, 2012; Strizhevskaya et al., 2012), i.e., the count of the occurence of queries, which assumes that the current and future query popularity is the same as past observed query popularity. Although this approach results in satisfactory QAC performance on average, it is far from optimal since it fails to take strong clues from time, trend and user-specific context into consideration while such information often influences the queries most likely to be submitted.

Previous work (Shokouhi, 2011; Shokouhi and Radinsky, 2012) shows that time-series analysis techniques can be used for classifying seasonal queries and forecasting their future popularity, which suggests that these schemes should be embedded into popularity-based query completion and query categorization approaches. In this thesis we continue to investigate the characteristics of query popularity and develop a time-sensitive query auto completion approach, which is further combined with a model that takes user's personal search history, both short-term (in the current session) and long-term (in previous logs), into consideration. Our proposal generally can promote user's intended query, returning the correct query candidate early in the QAC list. Moreover, we extend our model to specifically deal with long-tail prefixes by optimizing the contributions from query popularity and user-specific context.

Following popularity-based query auto completion approaches (Bar-Yossef and Kraus, 2011; Shokouhi and Radinsky, 2012; Strizhevskaya et al., 2012), where counting queries follows a strict query matching policy, in this thesis, we argue that the popularity of

queries that are similar to a query completion is also important for ranking the original query completions. Thus we consider the contributions from so-called homologous queries when ranking the initial query candidates. Homologous queries typically include queries with the same terms but ordered differently or queries that expand the original query candidate. In addition, we take features from computing semantic similarity between query terms into consideration as we state that users are prone to combine semantically related terms when generating queries. Based on those newly developed features, a learning-to-rank approach is directly implemented to generate a ranking model. Our analysis reveals that features of semantic relatedness and homologous queries are important and do indeed help boost QAC performance.

The third focus of the work in this thesis is on diversifying query auto completion (D-QAC), which has not been studied so far. Previous work on query auto completion mainly centers around returning a list of completions to users, aiming to push queries that are most likely intended by the user to the top positions but ignoring the redundancy among the query candidates in the list. Thus, semantically related queries matching the input prefix are often returned together. This may push valuable suggestions out of the list, given that only a limited number of candidates can be shown to the user. Hence, this may result in a less than optimal search experience. Unlike the task of search result diversification (Bache et al., 2013; Carbonell and Goldstein, 1998; Dang and Croft, 2012; Radlinski and Dumais, 2006; Zhai et al., 2003), where the main focus is centered around diversifying web search results, we aim to return correct query completions early in a ranked list of candidate completions and at the same time reduce the redundancy among query auto completion candidates. We develop a greedy query selection approach that predicts query completions based on the current search popularity of candidate completions and on the intents of previous queries in the same search session. We quantify the improvement of our greedy query selection model against a state-of-the-art baseline in terms of well-known metrics used in query auto completion and diversification.

To cater for a user's specific information needs, personalized query auto completion strategies have been investigated by taking their search history and their user profile into account. Such methods personalize the list of query completions in the same manner. However, it is unclear whether personalization is consistently effective to query auto completion under different search contexts. Consequently, our final focus in this thesis is on selectively personalizing query auto completion. Based on a lenient personalized QAC strategy that basically encodes the ranking signal as a trade-off between query popularity and personal search context, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model to study such a trade-off. In particular, we predict an effective trade-off in each case based on a regression model, where the typed prefix, the clicked documents and the preceding queries in session are considered for weighing personalization in the SP-QAC model. We find that personalization can be selectively embedded into a QAC approach rather than uniformly implemented in a QAC framework.

1.1 Research Outline and Questions

The broad question that motivates the research for this thesis is: *How can we improve the performance of query auto completion (QAC) in information retrieval?* Individual

components towards solving this problem already exist (see Chapter 2 for an overview), but other aspects, such as how to incorporate strong clues from time, user context or semantics, have not yet been well investigated. This thesis aims to close some of these gaps, contributing new scenarios for query auto completion in the field of information retrieval.

We start our investigation by focusing on combining information from both time-sensitive characteristics and user-specific features for query auto completion. In previous work, time-sensitive query auto completion (QAC) models and user-specific QAC approaches have been developed separately. Both types of QAC methods lead to important improvements over models that are neither time-sensitive nor personalized. We first propose the time-sensitive QAC models, i.e., λ-TS-QAC and λ^*-TS-QAC, that employ a fixed trade-off λ and an optimal trade-off λ^*, respectively, to control the contribution of recent trends and periodic signals when predicting query's future popularity. Given that the seasonal change (Shokouhi, 2011) and the recent trend (Whiting and Jose, 2014) of the search popularity of queries can be used for predicting a query's future popularity, we first aim to understand how these two parts can be integrated to improve the accuracy of query popularity prediction using time-sensitive information. We compare the prediction results generated by various models to answer the following questions:

RQ1 As a sanity check, what is the accuracy of query popularity prediction generated by various models?

RQ2 How do our time-sensitive QAC models (λ-TS-QAC and λ^*-TS-QAC) compare against state-of-the-art time-sensitive QAC baselines?

In answering these two research questions, we find that our prediction method, based on the periodicity and on the recent trend of query popularity, can produce accurate predictions of query popularity and perform better in terms of Mean Absolute Error (MAE) and Symmetric Mean Absolute Percentage Error (SMAPE) than other aggregation- and trend-based prediction baselines. Based on predicted query popularity, our proposed time-sensitive QAC model achieves better performance in terms of Mean Reciprocal Rank (MRR) than previous baselines.

After that, we propose a hybrid QAC model λ^*-H-QAC that considers both time-sensitivity and personalization to compare with an n-gram based hybrid model λ^*-H$_G$-QAC. Besides, an extension of λ^*-H-QAC, λ^*-H$'$-QAC, is proposed to deal with long-tail prefixes, i.e., unpopular prefixes, by optimizing the contributions from the predicted query popularity and from the user-specific context. To verify the effectiveness of proposed QAC models, we particularly answer the following research questions:

RQ3 Does λ^*-H-QAC outperform time-sensitive QAC methods, e.g., λ^*-TS-QAC)?

RQ4 How does λ^*-H-QAC compare against personalized QAC method using n-gram based query similarity?

RQ5 How does λ^*-H-QAC compare against λ^*-H$_G$-QAC?

RQ6 How does λ^*-H$'$-QAC compare against λ^*-H-QAC on long-tail prefixes? And on all prefixes?

Our experimental results show that, after integrating the user-centered search context with our time-sensitive QAC model, our proposal, i.e., a hybrid QAC approach, further boosts the ranking performance of query completions.

Analyzing previously developed query auto completion methods, we find that most of today's QAC models rank candidate queries by popularity (i.e., frequency), and in doing so they tend to follow a strict query matching policy when counting the queries as we pointed out in the following section. That is, they ignore the contributions from so-called homologous queries, i.e., queries with the same terms but ordered differently, or queries that expand the original query. Importantly, homologous queries often express a remarkably similar search intent. Moreover, today's QAC approaches often ignore semantically related terms. We argue that users are prone to combine semantically related terms when generating queries. To address this shortcoming, based on a learning-based QAC model L2R-U which extract features from user behaviors (Jiang et al., 2014b), we propose several learning to rank-based QAC approaches, where, for the first time, features derived from predicted popularity, homologous queries and semantically related terms are introduced, respectively.

In particular, we consider: (1) the observed and predicted popularity of query completions, which results in the L2R-UP model; (2) the observed and predicted popularity of homologous queries for a query candidate, which results in the L2R-UPH model; (3) the semantic relatedness of pairs of terms inside a query and pairs of queries inside a session, which results in the L2R-UPS model; and (iv) all these newly proposed features, which results in the L2R-ALL model. Regarding these new models, we address the following research questions:

RQ7 Do the features that describe the observed and predicted popularity of a query completion help boost QAC performance without negatively impacting the effectiveness of user behavior related features proposed in (Jiang et al., 2014b)? That is, how does L2R-UP compare against L2R-U?

RQ8 Do semantic features help improve QAC performance? That is, how does L2R-UPS compare against L2R-UP?

RQ9 Do homologous queries help improve QAC performance? That is, how does L2R-UPH compare against L2R-UP?

RQ10 How does L2R-UPS compare against L2R-UPH? What is the performance gain, if any, if all features are added for learning (L2R-ALL)?

RQ11 What are the principal features developed here for a learning to rank based QAC task?

Our experimental analysis reveals that features of semantic relatedness and homologous queries are important and they do indeed help boost QAC performance. In other words, query terms are not randomly combined when a searcher formulates a query. Semantically close terms or queries are likely to appear in a query or in a session, respectively.

We then turn to a practical issue of query auto completion in a web search setting. In this setting, previous work on query auto completion is mainly centered around returning a list of completions to users, aiming to push queries that are most likely intended by the user to the top positions but ignoring the redundancy among the query candidates in the list. Thus, semantically related queries matching the input prefix are often

returned together. This may push valuable suggestions out of the list, given that only a limited number of candidates can be shown to the user, and hence, this may result in a less than optimal search experience. To address this problem, we consider the task of diversifying query auto completion (D-QAC), which aims to return the correct query completions early in a ranked list of candidate completions and at the same time reduce the redundancy among query auto completion candidates. In particular, we propose a series of greedy query selection (GQS) models, i.e., GQS$_{MPC+AQ}$, GQS$_{MSR+AQ}$, GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$, corresponding to a GQS model that first selects the most popular completion and use all previous queries in a session as search context, that first selects the most similar completion and use all previous queries in session as search context, that first selects the most popular completion and use only the last preceding query in session as search context and that first selects the most similar completion and use only the last preceding query in session as search context, respectively. For this new D-QAC task, we try to answer the following questions for this problem:

RQ12 Do our greedy query selection (GQS) models beat the baselines for diversifying query auto completion task in terms of metrics for QAC ranking (e.g., MRR) and for diversification (e.g., α-nDCG)?

RQ13 How does the choice of selecting the first query to be included in the QAC result list impact the performance in diversified query auto completion of our GQS model?

RQ14 What is the impact on diversified query auto completion performance of our GQS model of the choice of search context, i.e., choosing all previous queries in a session or only the last preceding query?

RQ15 What is the relative D-QAC performance of our QAC models when evaluated using a side-by-side comparison?

RQ16 What is the sensitivity of our GQS model? In particular, how is the performance of our GQS model influenced by, e.g., the number of returned query auto completion candidates, namely a cutoff N, the number of latent features used in BPMF k_f and a trade-off λ controlling the contribution of search popularity and search context when modeling the closeness of query completion to search intent?

The proposed GQS models predict query completions based on the current search popularity of candidate completions and on aspects of previous queries in the same search session. The popularity of completion candidates at query time can be directly aggregated from query logs. However, query aspects are implicitly expressed by previous clicked documents in the search context. To determine the query aspect, we categorize clicked documents of a query using a hierarchy based on the open directory project. Bayesian probabilistic matrix factorization (BPMF) is applied to derive the distribution of queries over all aspects. Our experimental results show that our greedy query selection model can remove redundant query completions from the returned QAC lists without affecting the ranking performance of query completions.

Finally, we turn to the question how to incorporate personalization effectively in a generic QAC approach. We assume that the weight of personalization in a hybrid QAC model, which considers both the search popularity and search context when ordering the

query completions, can be non-uniformly assigned. Based on a lenient personalized QAC strategy that basically encodes the ranking signal as a trade-off between query popularity and search context, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model to study such a trade-off. In particular, we predict an effective trade-off in each case based on a regression model, where the typed prefix, the clicked documents and the preceding queries in session are considered for weighing personalization in QAC. The research questions addressed by our preliminary study are:

RQ17 Does selective personalization scheme help improve the accuracy of ranking query completions in a generic personalized QAC approach?

RQ18 How is the performance of proposed SP-QAC model under various inputs to the regression model for weighing personalization in a QAC task?

We demonstrate that the typed prefix yields the most benefits for weighing personalization in a QAC model and that the preceding queries contributes more than the click information. This work makes an important step towards unifying prior work on personalized QAC by studying when and how to incorporate personalization in QAC. We will continue to explore other sources for investigating how to best personalize query auto completion, e.g., a user's dwell time on clicked results and their long-term search history. In addition, it is interesting to zoom in on each particular user to know whether he is at all susceptible to personalization in QAC.

1.2 Main Contributions

In this section, the main contributions of this thesis are summarized as follows.

- **A prefix-adaptive and time-sensitive approach for personalized query auto completion.** As previous time-sensitive and user-specific query auto completion methods have been developed separately, yielding significant improvements over methods that are neither time-sensitive nor personalized, we propose a hybrid query auto completion (QAC) method that is both time-sensitive and personalized. We extend it to handle long-tail prefixes, which we achieve by assigning optimal weights of the contribution from time-sensitivity and personalization. Using real-world search log datasets, we first return the top N query completions ranked by predicted popularity as estimated from popularity trends and cyclic popularity behavior; we rerank them by integrating similarities to a user's previous queries (both in the current session and in previous sessions). Our method outperforms state-of-the-art time-sensitive QAC baselines, achieving total improvements of between 3% and 7% in terms of mean reciprocal rank (MRR). After optimizing the weights, our extended model achieves MRR improvements of between 4% and 8%.
- **A learning to rank based approach for query auto completion.** We propose a learning to rank-based QAC approach, where, for the first time, features derived from homologous queries and semantically related terms are introduced. In particular, we consider: (1) the observed and predicted popularity of homologous queries for a query candidate; and (2) the semantic relatedness of term pairs inside a query

and of query pairs inside a session. We quantify the improvements of the proposed new features using two large-scale real-world query logs and show that the mean reciprocal rank and the success rate at the top K can be significantly improved by up to 9% over state-of-the-art QAC models.

- **A greedy query selection approach for diversifying query auto completion.** We consider the task of diversifying query auto completion (D-QAC), which aims to return the correct query completions early in a ranked list of candidate completions and at the same time reduce the redundancy among query auto completion candidates. A greedy query selection approach is proposed to deal with the new D-QAC task, and finally we quantify the improvement of our greedy query selection model against a state-of-the-art baseline using two large-scale real-world query logs and show that it beats the baseline in terms of well-known metrics used in query auto completion and diversification. In addition, we implement a side-by-side experiment to verify the effectiveness of our outcome.
- **A selectively personalizing query auto completion approach.** We propose a model for Selectively Personalizing Query Auto Completion (SP-QAC), re-ranking the top N query completions by popularity, where personalization is individually weighted when being combined with ranking signals from search popularity. In particular, we study the following factors for weighing personalization: the typed prefix for which we recommend query suggestions, the clicked documents for inferring user's satisfaction and the topic changes of preceding queries in session for detecting search intent shifts. The experimental results reveal that the SP-QAC model, which selectively outweighs or depresses the contribution of personalization in a generic QAC approach, outperforms a traditional non-personalization QAC approach and a uniformly personalized QAC approach with a fixed trade-off controlling the contribution of search popularity and search context.

1.3 Thesis Overview

This section gives an overview of the content of each chapter of this thesis. Besides the current Introduction chapter, we have two chapters that summarize the main work related to query auto completion and the main methodology for evaluating the performance of various QAC algorithms, respectively. Following that, four research chapters detail our contributions in this thesis and the conclusion chapter highlights our findings.

Chapter 2: Background. In this chapter, we review previous work on query auto completion in the field of information retrieval. In particular, we discuss the motivations for query auto completion mainly in web search and then point out some recent advances as well as potential research directions in the field of query auto completion.

Chapter 3: Experimental methodology. In this chapter, we detail the problem formulation of query auto completion used throughout this thesis, describe the public datasets for experiments and introduce the experimental setup that forms the basis of the empirical evaluations. In addition, well-known metrics for QAC ranking,

e.g., MRR (Mean Reciprocal Rank) and SR (Success Rate), and for QAC diversification, e.g., ERR- IA (Intent-aware Expected Reciprocal Rank) and α-nDCG (Normalized Discounted Cumulative Gain), are introduced in detail.

Chapter 4: Prefix-adaptive and time-sensitive personalized query auto completion. In this chapter, we adopt a combination of the two aspects of the QAC problem, where time-series analysis is used to predict a query's future frequency. To understand a user's personal search intent, we extend our time-sensitive QAC method with personalized QAC, which infers the similarity between current requests and preceding queries in a current search session and previous search tasks at the character level. We verify the effectiveness of our proposed hybrid model λ^*-H-QAC on two datasets, showing significant improvements over various time-sensitive QAC baselines. In addition, we adjust the model specific for long-tail prefixes, resulting in clear improvements, especially for short prefixes.

Chapter 5: Learning from homologous queries and semantically related terms for query auto completion. In this chapter, we follow a supervised learning to rank approach to address the problem of ranking query auto completion (QAC) candidates. We develop new features of homologous queries and semantic relatedness of terms inside a query and of pairs of terms from a query candidate and from earlier queries in the same session. Our analysis reveals that features of semantic relatedness and homologous queries are important and do indeed help boost QAC performance.

Chapter 6: Diversifying query auto completion. In this chapter, we propose a greedy query selection (GQS) model to address the query auto completion diversification task and use the ODP (Open Directory Project) taxonomy to identify aspects of URLs, based on which we then assign query aspects via clickthrough data derived from query logs. We experimentally investigate the diversified query auto completion performance of our GQS models under various settings. Our results show that the GQS model performs best when starting with the semantically most closely related query completion and using only the last preceding query in a session as the search context. This finding indicates that query aspects are commonly shared by successive queries in a session and may be changed especially in long sessions with multiple queries. In addition, we find that our GQS model performs better when more query completions are fed to it.

Chapter 7: Selectively personalizing query auto completion. In this chapter, we propose a selectively personalized approach for query auto completion (QAC). In particular, our model predicts whether a specific prefix should be outweighed on personalization when ranking the query completions. We explore several factors that influence the weight of personalization in a generic QAC model, such as the typed prefix, the clicked documents and the preceding queries in session. We find that the typed prefix yields the most benefits for weighing personalization in QAC reranking and that the preceding queries contributes more than the click information. This work makes an important step towards unifying prior work on personalized QAC by studying when and how to incorporate personalization in QAC.

Chapter 8: Conclusion. We go back to the research questions introduced in this chapter and provide detailed answers. Finally, we identify future directions following the work in this thesis.

Chapter 2 serves as an overview to the related study in the research chapters and can be read if additional insight in the field is required. Chapter 3 provides some necessary descriptions on the test collections and evaluation metrics that are used for the experiments in the research chapters and gives additional details about the baseline query auto completion approaches. The research chapters, i.e., Chapters 4 to 7, can be read individually, as the content of these chapters is independent of that of other research chapters. Finally, reading only this introduction chapter and the conclusion in Chapter 8 gives a brief summary of the whole thesis, and provides insights to the research questions mentioned early in this chapter.

1.4 Origins

The following publications form the basis of chapters in this thesis.

- **Chapter 2** is based on (Cai and de Rijke, 2016b) "Query Auto Completion in Information Retrieval: A Survey," submitted to *Foundations and Trends in Information Retrieval, 2016*. Both authors have substantial contributions to the overall synthesis of the material. The writing was mostly done by Cai.

- **Chapter 4** is based on (Cai et al., 2014b) "Time-sensitive Personalized Query Auto Completion," published at *CIKM 2014* and its extension (Cai et al., 2016a) "Prefix-adaptive and Time-sensitive Personalized Query Auto Completion," accepted subject to minor revisions by *IEEE Transactions on Knowledge Discovery and Engineering, 2016*. All authors contributed to the design of the algorithms and of the experiments. The experiments and analysis were carried out mostly by Cai. The writing was mostly done by Cai.

- **Chapter 5** is based on (Cai and de Rijke, 2016a) "Learning from Homologous Queries and Semantically Related Terms for Query Auto Completion," to appear in *Information Processing & Management, 2016*. Both authors contribute to the ideas and experimental designs. The experiments and analysis were carried out mostly by Cai. The writing was mostly done by Cai.

- **Chapter 6** is based on (Cai et al., 2016b) "Diversifying Query Auto Completion," to appear in *ACM Transactions on Information Systems, 2016*. All authors contributed to the design of the algorithms and to the experimental design. The experiments and analysis were carried out mostly by Cai and Reinanda. The writing was mostly done by Cai.

- **Chapter 7** is based on (Cai and de Rijke, 2016c). "Selectively Personalizing Query Auto Completion," to appear at *SIGIR 2016*. Both authors contributed to the design of the algorithms and to the experimental design. The experiments and analysis were carried out mostly by Cai. The writing was mostly done by Cai.

In addition, this thesis draws on insights and experiences from the following materials:

- (Cai et al., 2014a) "Personalized Document Re-ranking Based on Bayesian Probabilistic Matrix Factorization," published at *SIGIR 2014*.
- (Cai et al., 2016c) "Behavior-based Personalization in Web Search," to appear in *Journal of the Association for Information Science and Technology, 2016*.
- (Cai and de Rijke, 2015b) "Personalized Web Search Based on Bayesian Probabilistic Matrix Factorization," published at *RuSSIR 2015*.
- (Cai and de Rijke, 2015a) "Time-aware Personalized Query Auto Completion," published at *DIR 2015*.
- (Liang et al., 2015) "Efficient Structured Learning for Personalized Diversification," accepted subject to major revisions by *IEEE Transactions on Knowledge Discovery and Engineering, 2015*.

2 Background

In this chapter, we provide a formal definition of the query auto completion (QAC) problem in §2.1, which is followed by section §2.2 discussing probabilistic QAC models and a section §2.3 describing learning-based QAC models.

2.1 Problem Formulation

Formulating queries and, especially, formulating effective queries is an important ingredient of the search experience. Since 2008, Google has offered a service named "Google Suggest," which provides users with query completions shown below the search box as a drop-down menu while they type, in real time.[1] To a certain degree, the completions are based on exploring what others are searching for. From Google's official blog,[2] the Google Suggest project was already launched in 2004. In 2011, Bar-Yossef and Kraus (2011) referred to this functionality as *query auto completion* (QAC) and proposed a straightforward approach to deal with the task of generating query completions, i.e., the well-known *most popular completion* (MPC) approach, which is based on the search popularity of queries matching the prefix entered by the user.

In essence, the QAC problem can be view as a ranking problem. Given a prefix, possible query completions are ranked according to a predefined criterion, and then some of them are returned to the user. Typically, a pre-computed auto completion system is required for generating query completions corresponding to each specific prefix in advance; it stores the associations between prefixes and query completions in an efficient data structure, such as prefix-trees, that allows efficient lookups by prefix matching. This index is similar to an inverted table storing a mapping from query to documents in an information retrieval system. Figure 2.1 illustrates a basic QAC framework. As users' queries and interactions can be recorded by search engines, this kind of data is used for generating an index table offline; it captures the relationships between prefixes and queries. When a user enters a prefix in the search box, based on the pre-computed table, a list of query completions will be retrieved. Based on further re-ranking using signals determined at query time, e.g., time, location and user behavior, the user will receive a final list of query completions. In deployed systems, the list of completions typically has

[1] `http://searchengineland.com/how-google-instant-autocomplete-suggestions-work-62592`

[2] `http://googleblog.blogspot.nl/2004/12/ive-got-suggestion.html`

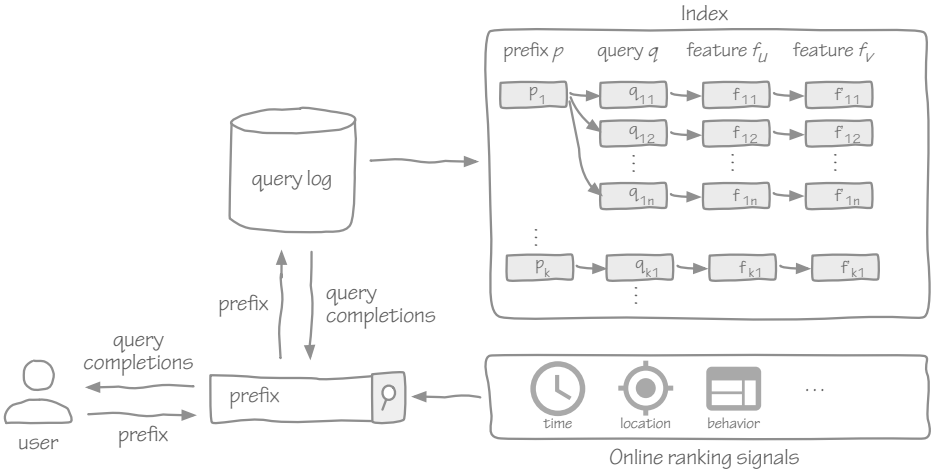


Figure 2.1: A basic QAC framework.

a limited length, e.g., at most 4 for Baidu, 8 for Bing, 4 for Google, 10 for Yahoo and Yandex.

Next, we introduce the problem of query auto completion more formally. Let p denote a prefix entered by a user u, i.e., a string of characters. Let Q_I denote a set of complete queries that extend p; it is helpful to think of Q_I as the set of initially retrieved completions (similar to top-k retrieval in standard document retrieval). The query auto completion problem is to find a ranking Q_S of the queries in Q_I, with $|Q_S| = N$, where $N > 0$ is a given cutoff denoting the number of top ranked elements of Q_I to be included in Q_S, such that

$$\Phi(Q_S) = \sum_{q \in Q_S} \phi(q), \text{ where } \phi(q) = \begin{cases} \frac{1}{\text{rank of } q \text{ in } Q_S}, & \text{if } q = q' \\ 0, & \text{if } q \neq q' \end{cases} \tag{2.1}$$

is maximized, where q' is a query that is finally submitted by the user u. Algorithmic solutions to the QAC problem aim to predict the user's intended query and then return it early in the candidate list Q_S. We refer to the items in the list Q_S as *query completions* or simply *completions*.

So far, several approaches have been proposed in the literature for the query auto completion task and a brief comparison of current approaches to ranking candidate query completions is proposed by Di Santo et al. (2015). We classify existing approaches to query auto completion into two broad categories, i.e., *probabilistic models* and *learning-based models*. The probabilistic approaches seek to compute a probability, i.e., a ranking score, for each query candidate that indicates how likely it is that this query would be issued. In contrast, learning-based approaches, based on a learning algorithm, aim to extract dozens of reasonable features to capture the characteristics of each query candidate. Each of these two categories of QAC approaches, can be split it into two groups: time-related and user-centered. In this manner we arrive at a two-by-two grid, which we adopt to organize the previous work summarized in this chapter; see Table 2.1.

Table 2.1: Representative query auto completion approaches in the literature.

QAC strategy	Representative work	
	Time-related	User-centered
Probabilistic	(Cai et al., 2014b; Shokouhi and Radinsky, 2012; Whiting and Jose, 2014)	(Bar-Yossef and Kraus, 2011; Li et al., 2015)
Learning-based	(Cai and de Rijke, 2016a), Chapter 4	(Jiang et al., 2014b; Li et al., 2014; Mitra, 2015; Shokouhi, 2013)

2.2 Probabilistic QAC Approaches

In this section, we describe representative probabilistic query auto completion approaches from the literature that consider time-related aspects (see §2.2.1) and personalization (see §2.2.2) to compute probabilities for ranking query completions. Basically, these approaches aim to compute a probability $P(q_c \mid t, u)$ of submitting a query completion q_c, given, for instance, the time t and the user u.

2.2.1 Time-sensitive Query Auto Completion Models

A straightforward approach to rank query completions is to use Maximum Likelihood Estimation (MLE) based on the past popularity (i.e., the frequency) of queries (Bar-Yossef and Kraus, 2011). Bar-Yossef and Kraus (2011) refer to this type of ranking as the most popular completion (MPC) model:

$$MPC(p) = \underset{q \in C(p)}{\operatorname{argmax}}\, w(q), \text{ where } w(q) = \frac{f(q)}{\sum_{q_i \in Q} f(q_i)}, \tag{2.2}$$

where $f(q)$ denotes the number of occurrences of query q in search log Q, i.e., its frequency, and $C(p)$ is a set of query completions that start with prefix p. In essence, the MPC model assumes that the current query popularity distribution will remain the same as that previously observed, and hence completions are ranked by their past popularity in order to maximize QAC effectiveness for all users on average. However, query popularity may change over time. Accordingly, the ranking of query completions must be adjusted to account for time-sensitive changes, which could be addressed by limiting the aggregation period of past query log evidence to increase the temporal sensitivity of QAC.

As the web increasingly becomes a platform for real-time news search and media retrieval, time plays a central role in information retrieval activities and more people are turning to find out about unpredictable, emerging and ongoing events and phenomena. For instance, a substantial volume of daily top queries are issued about the results related to up-to-date information of recent or ongoing events (Whiting and Jose, 2014). In addition, 15% of the daily queries to an industrial web search engine have never been

seen before.[3] A substantial proportion of these queries are attributed to fresh events and real-time phenomena, rather than the long-tail of very uncommon queries (Whiting et al., 2013). QAC approaches that are based on aggregating long-term historic query-logs are not sensitive to very fresh real-time events, which may result in poor performance for ranking query completions as newly popular queries will be outweighed by long-term popular queries, especially for short prefixes (e.g., consisting of 1 or 2 characters).

Inspired by the interesting findings from Google trends, Shokouhi (2011) focus on detecting seasonal queries using time-series analysis. They depict how a query's monthly popularity cycle can be treated as time-series and decomposed by Holt-Winters additive exponential smoothing (Goodwin, 2010) into three main components, i.e., level ($\mathcal{L}$), trend ($\mathcal{T}$) and season ($\mathcal{S}$):

$$\begin{cases} \mathcal{L}_t = \alpha \cdot (\hat{X}_t - \mathcal{S}_{t-s}) + (1-\alpha) \cdot (\mathcal{L}_{t-1} + \mathcal{T}_{t-1}) \\ \mathcal{T}_t = \beta \cdot (\mathcal{L}_t - \mathcal{L}_{t-1}) + (1-\beta) \cdot \mathcal{T}_{t-1} \\ \mathcal{S}_t = \gamma \cdot (\hat{X}_t - \mathcal{L}_t) + (1-\gamma) \cdot \mathcal{S}_{t-s}, \end{cases} \tag{2.3}$$

where α, β and γ are trade-offs in $[0, 1]$. The parameter t denotes the current time-span, and s specifies the length of the seasonal periodicity (e.g., 12 months). Furthermore, $\hat{X}_t$ represents the value of the data point at time t (e.g., in one month). Then, if the decomposed *season* component $\mathcal{S}$ and raw data $\hat{X}$ have similar distributions, the query is deemed to be a *seasonal* query. For instance, cyclic repeated events such as *Halloween* and *Christmas* happen every year, resulting in periodic spikes in the frequency of related queries (e.g., "*halloween costume*" and "*Christmas gift*"). Other queries, like "*sigir 2016*," are requests that target events that are close in time and thus are more likely to present a relatively obvious search burst than older alternatives (e.g., "*sigir 2010*"). Thus, for effective QAC, it is important to correctly identify recent popular queries and make sure that users who submit such a query do indeed have a temporal information need.

Building on the work of Shokouhi (2011), Shokouhi and Radinsky (2012) propose a time-sensitive approach for query auto completion. Instead of ranking candidates according to their past popularity, they apply time-series analysis and rank candidates according to the forecasted frequencies by modeling the temporal trends of queries. Unlike the ranking criterion specified in (2.2), this time-sensitive QAC approach can be formalized as follows:

$$TS(p, t) = \underset{q \in C(p)}{\operatorname{argmax}}\, w(q \mid t), \text{ where } w(q \mid t) = \frac{\hat{f}_t(q)}{\sum_{q_i \in Q} \hat{f}_t(q_i)}, \tag{2.4}$$

and where p is the input prefix, $C(p)$ represents a list of query completions matching the prefix p, and $\hat{f}_t(q)$ denotes the estimated frequency of query q at time t in the query log Q.

In practice, the future frequency of queries $\hat{y}_{t+1}$ can simply be forecast using the single exponential smoothing method (Holt, 2004) based on a previous observation y_t and a smoothed output $\bar{y}_{t-1}$ as follows:

$$\hat{y}_{t+1} = \bar{y}_t = \lambda \cdot y_t + (1-\lambda) \cdot \bar{y}_{t-1}, \tag{2.5}$$

[3] `http://www.google.com/competition/howgooglesearchworks.html`

where y_t and $\bar{y}_t$ denote the actually observed and smoothed values for the query frequency at time t, λ is a trade-off parameter ranging between 0 and 1, and $\hat{y}_{t+1}$ is the estimated future frequency of the query at time $t+1$. As the single exponential smoothing method cannot capture the *trend* and the *periodicity* of query popularity, the double and triple exponential smoothing methods (Holt, 2004) have been applied to forecast a query's future frequency (Shokouhi and Radinsky, 2012). For instance, the double exponential smoothing method extends the model in (2.5) as follows:

$$\begin{cases} \bar{y}_t = \lambda_1 \cdot y_t + (1-\lambda_1) \cdot (\bar{y}_{t-1} + F_{t-1}) \\ F_t = \lambda_2 \cdot (\bar{y}_t - \bar{y}_{t-1}) + (1-\lambda_2) \cdot F_{t-1} \\ \hat{y}_{t+1} = \bar{y}_t + F_t, \end{cases} \tag{2.6}$$

where λ_1 and λ_2 are smoothing parameters, and the forecast $\hat{y}_{t+1}$ at time $t+1$ depends on the variable F_t, which models the linear trend of the time-series at time t, and on the smoothed frequency $\bar{y}_t$ at time t. The triple exponential smoothing method adds a periodicity variable S_t to (2.6) as follows:

$$\begin{cases} \bar{y}_t = \lambda_1 \cdot (y_t - S_{t-T}) + (1-\lambda_1) \cdot (\bar{y}_{t-1} + F_{t-1}) \\ F_t = \lambda_2 \cdot (\bar{y}_t - \bar{y}_{t-1}) + (1-\lambda_2) \cdot F_{t-1} \\ S_t = \lambda_3 \cdot (y_t - \bar{y}_t) + (1-\lambda_3) \cdot S_{t-T} \\ \lambda_1 + \lambda_2 + \lambda_3 = 1 \\ \hat{y}_{t+1} = (\bar{y}_t + F_t) \cdot S_{t+1-T}, \end{cases} \tag{2.7}$$

where T indicates the periodicity, while λ_1, λ_2 and λ_3 are free smoothing parameters in $[0, 1]$. QAC methods based on time series analysis techniques consistently outperform baselines that use aggregated data; using more accurate query popularity predictions produced by time-series modeling leads to higher quality query suggestions (Shokouhi and Radinsky, 2012). Strizhevskaya et al. (2012) study actualization techniques for measuring prediction accuracy of various daily query popularity prediction models using query logs, which can be helpful to query auto completion.

Compared to query popularity prediction based on long-term query logs, short-range query popularity estimation has attracted more attention. Golbandi et al. (2013) develop a regression model to detect bursting queries for enhancing trend detection. By analyzing query logs, they seek to accurately predict what the most trending query items on the Web are. Various attempts have been made to make search trend predictions more accurate with low latency relative to the actual event that sparked the trend. Kulkarni et al. (2011) classify queries into different categories based on the changes in popularity over time and show that monitoring query popularity can reveal strong signals for detecting trends in query intent. In addition, Michail et al. (2004) develop a compressed representation for time-series and propose a model for detecting bursts in query frequencies. Instead of applying the time-series analysis technologies to predict a query's future popularity for query auto completion (Cai et al., 2014b; Shokouhi and Radinsky, 2012), Whiting and Jose (2014) propose several practical query completion ranking approaches, including: (i) using a sliding window to collect query popularity evidence from the past few days, (ii) generating the query popularity distribution in the stream of last N queries observed with a given prefix, and (iii) predicting a query's short-range popularity based on recently observed trends. The predictions produced by their last N query distribution

Table 2.2: Comparison of time-sensitive probabilistic query auto completion approaches.

Evidence	Requires	Representative references
Short-range	Recent search logs, i.e., search history in a recent period before querying time.	(Golbandi et al., 2013; Shokouhi and Radinsky, 2012; Whiting and Jose, 2014; Whiting et al., 2013)
Long-range	A large volume of search logs, i.e., search history covering a long-term period.	(Bar-Yossef and Kraus, 2011; Shokouhi, 2011)
Both	Short- and long-term search logs.	(Cai et al., 2014b; Shokouhi and Radinsky, 2012)

approach help achieve the best performance for query auto completion after learning the corresponding parameters online.

In addition to the aforementioned time-based clues for query auto completion, the relationship between query terms, such as semantic similarity, has also been studied in the setting of QAC. For instance, Chien and Immorlica (2005) demonstrate that queries with similar temporal patterns can be semantically related for query completion even in the absence of lexical overlap. Liu et al. (2008) introduce a unified model for forecasting query frequency, in which the forecast for each query is influenced by the frequencies predicted for semantically similar queries. These approaches are able to produce accurate popularity predictions of queries and thus boost the effectiveness of popularity-based query completion approaches. In Table 2.2, we compare the main approaches of time-sensitive probabilistic query auto completion in the field of information retrieval. Generally, more work has been published on the use of recent evidence than on the long-term search history for query auto completion. QAC models that combine both short-term observations (ongoing trend) and long-term observations (periodic phenomena) achieve the best performance in terms of query completion ranking, and QAC models only based on a long-term aggregated frequency are worse than those that only exploit more recent data. In addition, QAC models based on observed query popularity, e.g., (Bar-Yossef and Kraus, 2011; Whiting and Jose, 2014), basically assume that the current or future query popularity is the same as past query popularity. In contrast, models based on predicted query popularity, such as, e.g., (Cai et al., 2014b; Shokouhi and Radinsky, 2012), take strong temporal clues into consideration as such information often influences the queries to be entered.

2.2.2 User-centered Query Auto Completion Models

User activities recorded in browsers have been used extensively to create a precise picture of the information needs of users; this concerns both long-term browsing logs (Bennett et al., 2012; Liu et al., 2010b; Matthijs and Radlinski, 2011; Tan et al., 2006) and short-term search behavior (Collins-Thompson et al., 2011; Jiang et al., 2011; Shen et al., 2005; Ustinovskiy and Serdyukov, 2013). Such data has become an important resource

for search personalization (Dou et al., 2007; Sontag et al., 2012). In this section, we give an overview of user-centered QAC models, i.e., personalized query auto completion approaches, where information from a user's search context and information about their interactions with a search engine are exploited for query auto completion.

In most work mentioned so far, query completions are computed globally. That is, for a given prefix, all users are presented with the same list of candidates. But exploiting the user's personal context has led to increases in QAC effectiveness (Bar-Yossef and Kraus, 2011; Liao et al., 2011; Santos et al., 2013; Shokouhi, 2013). The search context can be collected from the current search session, e.g., the preceding queries. Bar-Yossef and Kraus (2011) treat the user's preceding queries in the current session as context and propose a context-sensitive query auto completion algorithm that produces the completions of the user input that are most similar to the context queries. In particular, they propose to output a set of completion q_c of prefix p whose vector representation v_q has the highest cosine similarity to the search context C as represented by a vector v_C:

$$q_c \leftarrow \underset{q \in completion(x)}{\operatorname{argmax}} \frac{v_q \cdot v_C}{||v_q|| \cdot ||v_C||}, \tag{2.8}$$

where $v_q \cdot v_C$ denotes the dot product of v_q and v_C. By doing so, a ranked list L_{NC} of query completions of prefix p consisting of the top k completions can be returned. At the same time, another list L_{MPC} consisting of the top k query completions can be produced based on query popularity. The final ranked list of query completions is then constructed by aggregating the two lists L_{NC} and L_{MPC} according to a hybrid score:

$$hybscore(q_c) \leftarrow \alpha \cdot NormSim(q_c) + (1 - \alpha) \cdot NormMPC(q_c), \tag{2.9}$$

where $0 \leq \alpha \leq 1$ is a free parameter determining the weight of the normalized similarity score $NormSim(q_c)$ relative to the weight of the normalized popularity score $NormMPC(q_c)$. This model works well in cases where the immediate search context is relevant to the user's intended query and, hence, query similarity can contribute. When the search context is irrelevant, this model has to rely on query popularity.

Another option is for the search context to be determined by a user's interactions. Li et al. (2015) pay attention to users' sequential interactions with a QAC engine in and across QAC sessions, rather than users' interactions at each keystroke of each QAC session. Through an in-depth analysis on a high-resolution query log dataset, they propose a probabilistic model that addresses the QAC task by capturing the relationship between users' sequential behaviors at different keystrokes. Zhang et al. (2015) study implicit negative feed back from a user's interactions with a search engine, and propose a novel adaptive model *adaQAC* that adapts query auto completion to users' implicit negative feedback about skipped query completions. This model is based on the assumption that top ranked but skipped query completions are not likely to be submitted. They propose a probabilistic model to encode the strength of the implicit negative feedback to a query completion q_c from a user u with personalization.

The user-centered QAC models that we have just discussed mainly explore information from the user's search context, e.g., previous queries, or the user's interactions while engaging with QAC, e.g., typing and skipping at each keystroke. This leads to a natural grouping of approaches as shown in Table 2.3. Search context-based QAC models do

Table 2.3: Comparison of user-centered probabilistic query auto completion approaches.

Category	Assumption	Issue	Representative work
Search context	Search context expresses similar query intent to that of the intended query.	Search intent may be changed, thus the search context is irrelevant to the intended query.	(Bar-Yossef and Kraus, 2011; Cai et al., 2014b)
Interaction	The transition between keystrokes is influenced by users' interactions in a QAC engagement.	Users may ignore the completed query candidates before submitting one.	(Li et al., 2015; Zhang et al., 2015)

not need a high-resolution query log dataset to record the user's sequential keystrokes for specific query completions; however, such data is mandatory in interaction-based QAC models. In addition, it is more difficult to reproduce interaction-based QAC models, e.g., (Li et al., 2015; Zhang et al., 2015) than QAC models based on search context, e.g., (Bar-Yossef and Kraus, 2011; Cai et al., 2014b), which are relatively easy to implement.

2.3 Learning-based QAC Approaches

The rise of learning to rank for QAC came on the heels of the introduction of a broad range of time series-based ranking principles in query popularity prediction (Cai et al., 2014b; Shokouhi and Radinsky, 2012) and an increased understanding of ranking principles based on user interactions with search engines (Li et al., 2015; Zhang et al., 2015). In a typical (supervised) feature-based learning to rank framework for document retrieval, the training data for a specific learning algorithm (whether pointwise, pairwise or listwise) consists of a set of query-document pairs represented by feature vectors associated with relevance labels, and the goal is to learn a ranking model by optimizing a loss function (Liu, 2003). In contrast, in a learning to rank-based framework for QAC, the input is a prefix $p_i = \{char_1, char_2, \ldots, char_{n_c}\}$, i.e., a string of n_c characters. Generally, there is a pool $Q_c(p_i)$ consisting of query completions that start with the prefix p_i and that could be issued after entering p_i. These completions can simply be sorted by popularity. Each completion in $Q_c(p_i)$ can be represented as a prefix-query pair feature vector v_q. In a L2R framework for QAC, the model is trained on prefix-query pairs with binary labels, i.e., "submitted" or "non-submitted," similar to the "relevance" label of query-document pairs in L2R for document retrieval. Table 2.4 provides a high level comparison of L2R for document retrieval with L2R for QAC.

As illustrated in Figure 2.2, learning-based QAC approaches mainly focus on extracting useful features to seek a correct prediction of the user's intended query. Such QAC models eventually learn a ranking function from the extracted features. We split learning-based QAC approaches into two groups according to where the features used

Table 2.4: Comparison of learning to rank for document retrieval (DR) vs. query auto completion (QAC). In a document retrieval task, for a given query q_i, each of its associated documents d can be represented by a feature vector $v_d = \Phi(d, q)$, where Φ is a feature extractor; $m^{(i)}$ is the number of documents associated with query q_i, i.e., D. In a QAC task, for a given prefix p_i, each of its auto completed query candidates q can be represented by a feature vector $v_q = \phi(p, q)$, where ϕ is a feature extractor; $n^{(i)}$ is the number of query candidates associated with prefix p_i), i.e., $Q_c(p_i)$.

DR	Input	Query q_i
	Ranking candidates	Documents $d \in D$
	Candidate features	TF, IDF, BM25, etc.
	Output	A ranked list of documents
	Ground truth	Multi-level relevance labels, i.e., 0, 1, 2
	Major metrics	MAP, P@K, NDCG@K, etc.
QAC	Input	Prefix p_i
	Ranking candidates	Queries $q \in Q_c(p_i)$
	Candidate features	Popularity, length, position in session, etc.
	Output	A ranked list of queries
	Ground truth	Binary labels: 1 for submitted query and 0 for the others.
	Major metrics	MRR, SR@K, etc.

come from, i.e., time-related characteristics in §2.3.1 and user interactions in §2.3.2.

2.3.1 Learning from Time-related Characteristics for Query Auto Completion

So far, time-related characteristics used for learning-based QAC approaches have not been well studied. In general, time-related characteristics used for learning mainly relate to popularity-based features from two sources: previous observations and future predictions. Such signals have been used in probabilistic query auto completion (Cai et al., 2014b; Whiting and Jose, 2014). Similar to (Whiting and Jose, 2014), query popularity is determined from recent days, for instance, from the recent 1, 2, 4, or 7-day interval preceding the prefix submission time as well as from the entire query log. In addition, the predicted query popularity for learning is generated by considering the recent trend as well as the cyclic behavior of query popularity. Such a scheme will be discussed in detail in Chapter 5.

2.3.2 Learning from User Interactions for Query Auto Completion

There is more published work on learning-based QAC approaches that focus on features based on user related characteristics than work than on features based on time-related characteristics. User centered features typically originate from one of two sources: (1) from user behavior in search sessions (Jiang et al., 2014b; Mitra, 2015; Shokouhi, 2013), which can capture user's search interests and how they change queries during

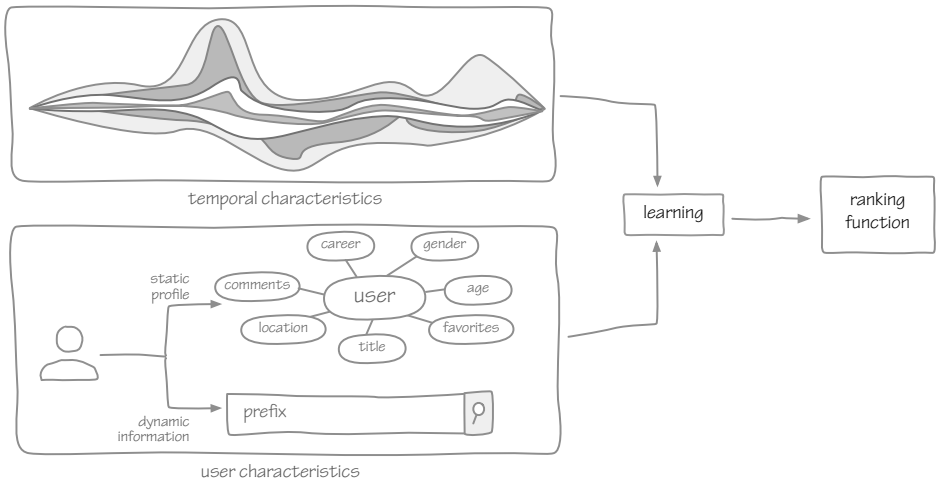


Figure 2.2: A general framework for learning-based query auto completion approaches.

search sessions; or (2) from high-resolution query logs (Li et al., 2014) that capture user feedback like query typing, skipping, viewing and clicking as feedbacks in an entire QAC process.

It has been observed that the popularity of certain queries may vary drastically across different demographics and users (Shokouhi, 2013). To better understand the role played by user profile and search context, the author presents a supervised framework for personalized ranking of query auto completions, where several user-specific and demographic-based features are newly extracted for learning. In particular, for user-specific context features, Shokouhi (2013) investigates the effectiveness for personalizing query auto completion of features developed from the search context consisting of both the short-term and long-term search history of a particular user. To generate the short-history features, the queries submitted previously in the current search session are used. To produce the long-history features, the whole search history of the particular user is considered. Then, similarity features of query completions can be measured by n-gram similarity to those queries in the context. Similar features have been used in previous work for personalized re-ranking web search results (Bennett et al., 2012; Teevan et al., 2011; Xiang et al., 2010) and suggested query candidates (Cao et al., 2008; Mei et al., 2008a). For demographic-based features, the information of a user's age, gender and location (e.g., zip-code) is considered. Based on a learning algorithm, e.g., (Burges et al., 2011), the experimental results show that demographic features such as location are more effective than others for personalizing query auto completion. In addition, adding more features based on users demographics and search history leads to further boosts in performance of personalized query auto completion.

Similarly, focusing on search context in the current search session, Jiang et al. (2014b) investigate the feasibility of exploiting the search context consisting of previously submitted queries to learn user reformulation behavior features for boosting the performance

of query auto completion. An in-depth analysis is conducted to see how the users reformulate their queries. Three kinds of reformulation-related features are considered, i.e., term-level, query-level and session-level features, which can capture clues as to how users change their preceding queries along the query sessions. Compared to traditional context-aware QAC methods that directly model term similarities or the query dependencies, this learning-based QAC approach tries to learn how users change preceding queries, i.e., user reformulation behavior, along query sessions, which results in significant improvement over the performance of start-of-the-art context-aware query completion or suggestion approaches (Bar-Yossef and Kraus, 2011; He et al., 2009; Liao et al., 2011).

Mitra (2015) builds on the insight that search logs contain lots of examples of frequently occurring patterns of user reformulations of queries. They represent query reformulations as vectors, which is achieved by studying distributed representations of queries learnt by a deep neural network model, like the convolutional latent semantic model (CLSM) (Shen et al., 2014). By doing so, queries with similar intents are projected to the same neighborhood in the embedding space. Based on the distributed representation of queries learnt by the CLSM model, topical similarity features measured by the cosine similarity between the CLSM vectors corresponding to query completions and the previous queries in the same search session are computed and used as features for learning in the QAC ranking model. In addition to the CLSM topical similarity features, the CLSM reformulation features, represented by low-dimensional vectors which map syntactically and semantically similar query changes close together in the embedding space, are used in the supervised learning model. Interestingly, Mitra and Craswell (2015) focus on recommending query completions for the prefixes that are sufficiently rare as previous QAC approaches only center around recommending queries for prefixes that have been frequently seen by the search engine. In particular, a candidate generation approach using frequently observed query suffixes mined from historical search logs is proposed by adopting the CLSM (Shen et al., 2014) trained on a prefix-suffix pairs dataset. The training data for the CLSM model is generated by sampling queries in the search logs and segmenting each query at every possible word boundary. This approach provides a solution to recommend queries that could have never been seen before in the training period, an issue that cannot be addressed by previously proposed probabilistic algorithms, e.g., (Bar-Yossef and Kraus, 2011; Cai et al., 2014b, 2016a; Shokouhi and Radinsky, 2012; Whiting and Jose, 2014) and Chapter 4, nor by learning-based QAC approaches, e.g., (Cai and de Rijke, 2016a; Jiang et al., 2014b; Shokouhi, 2013) and Chapter 5, because these proposed models depend on the query popularity: only previously seen, sufficiently frequent queries can be returned.

High-resolution query logs capture more detail information than traditional logs, which typically contain the timestamp of a query, the anonymous user ID and the final submitted query. High-resolution logs may capture user interactions at every keystroke and associated system response. Based on observations from such rich data, Li et al. (2014) observe the phenomenon that, in QAC, users often skip query completion lists even though such lists contain the final submitted query and most of the clicked queries are concentrated at top positions of the query completion lists. Based on this observation, they propose two types of bias related to user behavior, i.e., a *horizontal skipping bias* and a *vertical position bias*, that are crucial for relevance prediction in QAC. First, the

Table 2.5: Comparison of learning-based query auto completion approaches.

Feature source	Strengths	Weaknesses
Time	Easy to implement; effective for fresh events.	Limited number of features; hard to predict unpopular completions; sensitive to the unpredictable events.
User	Dynamic intent by interaction; static interest by user profile.	Poor performance if intent shifts; hard to record a detailed QAC engagement; sensitive to user search habit; relatively difficult to reproduce.

horizontal skipping bias refers to the assumption that a query completion will not receive a click if the user does not stop and examine the suggested list of query completions, regardless of the relevance of the query completion. The vertical position bias relates to the exact position of query completion in the provided list that a query on higher rank tends to attract more clicks regardless of its relevance to the search intent.

Based on these observations, the authors propose a two-dimensional click model to interpret the QAC process with particular emphasis on these two types of behaviors. The phenomenon of position bias has previously been observed for traditional document retrieval (Craswell et al., 2008; Granka et al., 2004), and several solutions have been proposed in the setting of click models (Chapelle and Zhang, 2009; Chuklin et al., 2015b; Dupret and Piwowarski, 2008; Liu et al., 2010a; Richardson et al., 2007; Zhang et al., 2011; Zhu et al., 2010). Newly extracted features from high-resolution query logs, which accurately explain user behavior during QAC engagement, reveal that people tend to stop and look for query completions when they are typing at word boundaries, which is consistent with the findings in (Mitra et al., 2014).

Finally, we summarize the main strengths and weaknesses of the learning-based QAC approaches in Table 2.5. So far, user-centered QAC approaches have received more attention than time-related methods. We believe that this is due to the rich data recorded, based on which user's search intent can often be correctly predicted, resulting in good QAC performance.

2.4 Practical Issues

In this section, we discuss two practical issues related to query auto completion, i.e., efficiency in §2.4.1, presentation and interaction in §2.4.2.

2.4.1 Efficiency

Over the past couple of years, QAC has popped up in a range of search engines, like web search engines, e.g., Bing and Google, as well as social media platforms, e.g., Facebook and Twitter. It makes for an improved user experience (Zhang et al., 2015), but it can

be a challenge to implement efficiently. In many cases, a practical implementation of QAC requires the use of efficient algorithms and data structures for fast lookups. While efficiency issues in traditional keyword search (Ji et al., 2009; Li et al., 2011) and document retrieval (Bast et al., 2007; Francès et al., 2014) have been addressed in numerous publications, relatively few papers report on efficiency in query auto completion.

However, efficiency must be taken into consideration in a practical QAC framework as query completion systems should help users form queries when they enter a query prefix. After summarizing previous work related to efficiency issues in QAC, we find two main types of practical efficiency issues in QAC, i.e., computational efficiency and error-tolerant efficiency.

Computational efficiency in QAC

In this section, we discuss a practical issue related to QAC efficiency that has been studied in the literature that are centered around computational complexity, mainly focusing on reducing the precessing time for completing a query prefix.

Before be able to rank query completions, indexing hundreds of millions of distinct queries is required to guarantee a reasonable coverage corresponding to an input prefix. A trie is a simple data structure that supports a fast query completion lookup. Hsu and Ottaviano (2013) focus on the case where the completion set is so large that compression is needed to fit the data structure in memory. The authors present three trie-based data structures to handle such cases, i.e., Completion Trie, RMQ Trie and Score-Decomposed Trie. Each has different space vs. time vs. complexity trade-offs. Trie-based indexes of queries need to be maintained continuously. Such a scenario supports fast query completion lookups but needs too much memory. To solve the memory problem of tries, a ternary tree is a good solution where each trie node is represented as a tree-within-a-tree. Typically, the ternary tree is stored on a server before testing. Then, in a real-time QAC process, the client will send a prefix to the server to get query completions based on the constructed ternary tree. Matani (2011) present an algorithm that takes time proportional to $O(k \log n)$ for providing the user with the top k ranked query completions out of a corpus of n possible suggestions based on the ternary tree for completing the prefix of the query that user has entered so far.

As a trie-based data index scheme suffers an inherent problem that the number of prefixes is huge for the first few characters of the query and is exponential in the alphabet size, this will result in a slow query response even if the entire query approximately matches only few prefixes. In addition, the efficiency critically depends on the number of active nodes in the trie. To deal with this problem, Xiao et al. (2013) study a QAC problem that tolerates errors in user's input using edit distance constraints and propose a novel neighborhood generation-based algorithm, i.e., *IncNGTrie*, which can achieve up to two orders of magnitude speedup over existing methods for the query auto completion problem. Their proposed algorithm only needs to maintain a small set of active nodes in a trie, thus saving both space and time to process the query. In addition, they propose optimization techniques to remove duplicates and reduce the index size by merging common data strings and common subtrees.

Instead of keeping query completions in one trie, Kastrinakis and Tzitzikas (2010) propose to partition a trie into two or more subtries to make query completion services

more scalable and faster. In doing so, the query auto completion mechanism involves a forest of tries. In particular, each subtrie contains queries whose starting characters belong to a specific set of characters assigned to this trie. The key challenge is to partition the trie. A subtrie is created as a partition and gFigurerows until its size becomes greater than a desired partition capacity. After that, a new partition is created and so on. The experimental results show that the proposed partitioning scheme allows increasing the number of query completions to be hosted and speeding up their computation at real time.

Similar to the work on learning-to-rank for document retrieval (Liu, 2003), where the features of query-document paris are generated offline, learning-based QAC approaches (Cai and de Rijke, 2016a; Jiang et al., 2014b; Shokouhi, 2013) extract features of prefix-query completions in the training period before online testing for effective QAC performance. Figure 2.3 shows an example of typical data format for learning-to-rank in document retrieval and for learning-to-rank in query auto completion. As shown in Figure 2.3a, the features depict the relationships between the query and its associated documents, such as the query frequency in the document title, the word frequency in the document abstract, and the query frequency in the whole document. Such features are used for measuring the similarity between a query and a document. The labels in Figure 2.3a indicate the relevance between a query and a corresponding document. However, in a learning-based QAC framework as shown in Figure 2.3b, the features are introduced for measuring the possibility of submitting a query completion after entering the prefix and the label denotes whether the provided query completion is submitted or not after entering the prefix. For instance, the features related to the popularity of a query completion in the query log, the similarity of a query completion to the search context in session and the similarity of a query completion to a user's interest contained in their profile, etc., are commonly used in a learning-to-rank baed QAC approach. All these features are generated offline for training a ranking model. In an online test setting, the main time cost for ranking query completions are on the lookups for matching such query completions and on calculating a final ranking score based on a trained model, which generally incurs very limited time costs (Cai and de Rijke, 2016a; Jiang et al., 2014b); see also Chapter 5.

Error-tolerant efficiency in QAC

Another aspect related to efficiency of QAC is concentrated on efficient approaches that are resilient to typing errors. Just as there is a basic need for making the lookup operation tolerant to such typing errors, a QAC system needs to be error-tolerant when responding to an input query prefix. Chaudhuri and Kaushik (2009) take a first step towards addressing this problem by capturing input typing errors via edit distance and propose a naive approach of invoking an offline edit distance matching algorithm. Their proposal, a trie-based QAC strategy, is desirable to generate query completions per-character at a minimal cost, generally reporting an average per-character response time at a millisecond level and significantly outperforming an n-gram based algorithm.

As misspellings are commonly found in web search and more than 10% of search engine queries are misspelled according to (Cucerzan, 2007), Duan and Hsu (2011a) study the problem of online spelling correction for query completions, which involves not only

QueryID_1	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label
QueryID_1	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label
⋮					⋮				
QueryID_1	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label
	⋮					⋮			
QueryID_k	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label
QueryID_k	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label
⋮					⋮				
QueryID_k	DocID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_m	Value_m	Label

(a) An example of data format for learning-to-rank in document retrieval.

PrefixID_1	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label
PrefixID_1	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label
⋮					⋮				
PrefixID_1	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label
	⋮					⋮			
PrefixID_k	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label
PrefixID_k	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label
⋮					⋮				
PrefixID_k	QueryCompletionID	Feature_1	Value_1	Feature_2	Value_2	...	Feature_n	Value_n	Label

(b) An example of data format for learning-to-rank in query auto completion.

Figure 2.3: An example of data format for learning-to-rank in document retrieval and in query auto completion.

completing a typed prefix but also correcting parts of incorrectly typed prefix when providing query completions as the query is being entered. Specifically, the search engine responds to each keystroke with a list of query completions that best correct and complete the partial query. To deal with this online spelling correction for query completion task, they model search queries with a generative model, where the intended query is transformed into a potentially misspelled query through a noisy channel that describes the distribution of spelling errors, and finally propose to find the top spell-corrected query completions in real-time by adapting an A* search algorithm with various pruning heuristics to dynamically expand the search space efficiently. The experimental results demonstrate a substantial increase in the effectiveness of online spelling correction over existing techniques.

2.4.2 Presentation and Interaction

In search engines, the most common way that search results are displayed is as a vertical list of items summarizing the retrieved documents. The search results listings returned by search engines are often known as *search engine result pages (SERPs)*. In query auto completion (QAC), a similar way to present query completions responding a typed prefix can be implemented that as a user enters a query in a search box, matching query completions appear below the search box as a drop-down menu with the typed characters highlighted in each query completion.

In addition, users can interact with SERPs in the form of clicking, reading and skipping, etc., which provide valuable feedbacks indicating the relevance of documents to a query as well as user's search intent. In contrast, in a QAC process, user's interaction behaviors with a QAC system, e.g., typing, skipping and clicking, also can be recorded to generate a personal search pattern for a particular user, which could improve the pre-

diction accuracy of their intended query. Most research efforts are directed towards improving the accuracy of query completions, e.g., (Bar-Yossef and Kraus, 2011; Cai et al., 2014b; Shokouhi, 2013; Shokouhi and Radinsky, 2012), and generally pay less attention to the presentation issues.

In this section, we specifically discuss how to present QAC results and how people interact with QAC results during an information retrieval process. These two points could affect the QAC usage in practice.

Presenting Query Auto Completions Results

The representation for a retrieved document in SERPs is called a search *hit* (Kazai et al., 2011; Morris et al., 2008), which is sometimes referred to as *document surrogate* (Beitzel et al., 2005; Marchionini and White, 2007). Example of result presentation in document retrieval and in query auto completion are shown in Figure 2.4 and Figure 2.5, respectively. As shown in Figure 2.4, in a SERP, a brief summary of a relevant portion of

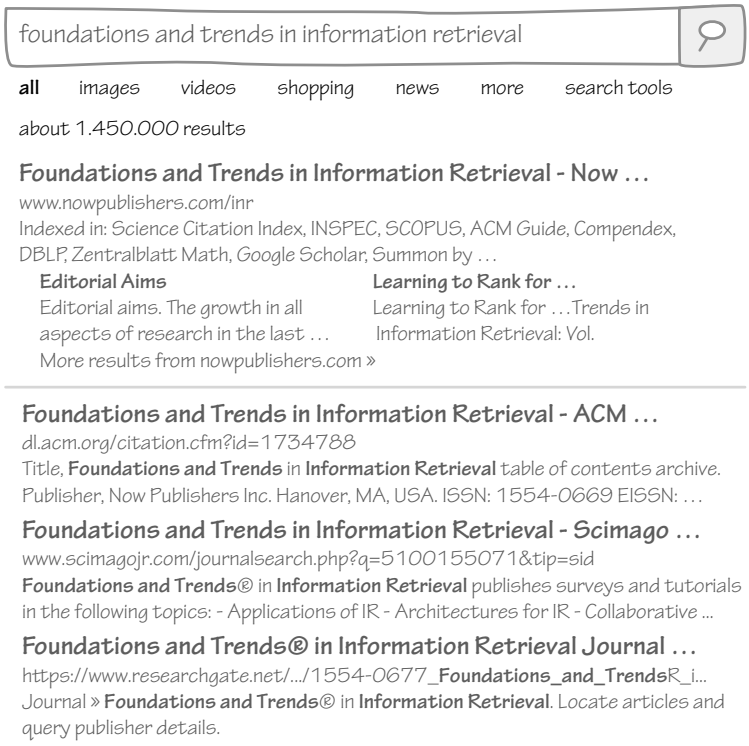


Figure 2.4: SERP resulting after querying "*foundation and trends in information retrieval*."

the document intended to help the user understand the primary object (Marchionini and White, 2007) is highlighted. The quality of the document surrogate has a strong effect

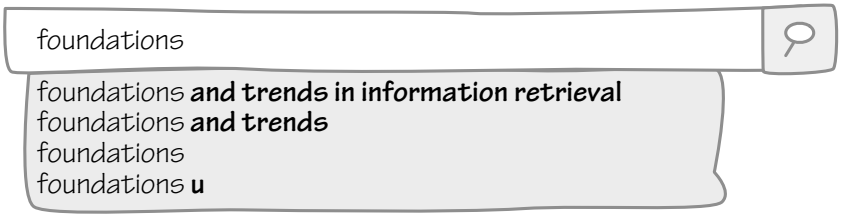


Figure 2.5: Query completions after entering "*foundations*."

on the ability of the searcher to judge the relevance of the document (Clarke et al., 2007). Even the most relevant document is unlikely to be selected if the title is uninformative or misleading. In addition, users are known to be biased towards clicking documents higher up in the rankings (Joachims et al., 2005). Such presentation issues in document retrieval have, for instance, been studied in (Joho and Jose, 2006, 2008).

In a QAC system, as shown in Figure 2.5, the query completions are listed below the search box and only a limited number of query completions are displayed to the user. In addition, the query completions that have been issued before can be highlighted with a different color. However, other information about the query completions is not provided. Similarly, the bias towards clicking query completions at top positions of the QAC list does exist (Li et al., 2014; Zhang et al., 2015). In fact, more than three quarters of clicks on query completions for desktop search are located within the top 2 positions of query completions lists and an even higher percentage is observed on mobile devices (Li et al., 2014). On the other hand, if this intended query is ranked outside the top 2 positions, only 13.4% of them will be clicked by users (Li et al., 2015). Hence, in a QAC system, the issues related to QAC result presentation should be taken into consideration for designing a QAC system as well as improving user's QAC experience.

Amin et al. (2009) focus on the presentation of query completions to see how it influences user's search performance. They implement user studies that investigate organization strategies, e.g., alphabetical order, group and composite, of query completions in a known-item search task, i.e., searching for terms taken from a thesaurus. They find that for a geographical thesaurus, a sensible grouping strategy, e.g., by country or by place type, that people can understand relatively easily is preferred. Regarding organization strategies of query completions, both group and composite interfaces help the user search for terms faster than alphabetical, and are generally easier to use and to understand than alphabetical. These findings provide insights on designing a better QAC interface for further improving user experience.

Based on the assumption that users would benefit from a presentation of query completions that are not only ordered by likelihood but also organized by a high-level user intent, Jain and Mishne (2010) specifically focus on organizing query completions by topic for web search, which is achieved by clustering and ordering sets of query completions. In particular, they first present a variety of unsupervised techniques to cluster query completions based on the information available to a large-scale web search engine, and then label each cluster by various scenarios. Finally, the clusters are ranked by minimizing an expected cost of locating query completions, e.g., time to read a cluster label

and time to scan a cluster.

Through a pilot user study, a few interesting and promising observations are obtained from (Jain and Mishne, 2010). For instance, users prefer query completions displayed in a clustered style rather than an unclustered presentation. The user effort in identifying the set of relevant completions can be substantially reduced by grouping query completions and labeling them, resulting in increased user's satisfaction. However, the tasks related to the clustering, e.g., selection of the queries to cluster, assignment of a label to each cluster, ordering the clusters as well as the query completions within each cluster, are still challenging. In general, this work opens new and promising directions towards improving the user experience by reducing the user effort in identifying the set of relevant query completions. These conclusions motivate follow-up work on diversifying query auto completion (Cai et al., 2016b); see Chapter 6.

Interaction with Query Auto Completion Results

In this section, we discuss users' interactions with the QAC results in some detail. With special devices, a user's explicit interactions during query auto completion engagement may be recorded. Such interactions provide valuable information for capturing a user's personal characteristics, e.g., typing habits and search interests.

Mitra et al. (2014) present the first large-scale study of user interactions with query auto completion. Their investigations reveal that lower-ranked query completions receive substantially less engagement, i.e., fewer clicks, than those ranked higher. In addition, users are most likely to submit a query by clicking a completion after typing about half of the query and in particular at word boundaries. Another factor that affects QAC engagement is related to the distance of query characters on the keyboard. For instance, a monotonic increase in the probability of QAC engagement with keyboard distance up to a distance of 4 keys can be observed, which demonstrates a clear relation between the position of the next character on the keyboard and the likelihood of QAC usage. Overall, such results appear to confirm typical intuitions of when users would like to use QAC. Furthermore, these results could be due to other attributes like user's typing habit, e.g., skipping and viewing completions. Such problems are studied in (Li et al., 2014; Zhang et al., 2015), based on a high-resolution query log dataset.

An in-depth eye-tracking study of user interactions with query auto completion in web search is performed by Hofmann et al. (2014). In their study, the participants' interactions are recorded using eye-tracking and client-side logging when they complete the assigned web search tasks. A strong position bias towards examining and using top-ranked query completions is identified, which is consistent with similar findings on user interactions with web search results (Chuklin et al., 2015a,b; Guo et al., 2009a; Yue et al., 2010). Due to this strong position bias, ranking quality does affect QAC usage, as evidenced by different behavioral patterns, which includes activities such as monitoring, skipping and searching for query completions. This work suggests that injecting user feedback into a QAC framework may be beneficial. The issue is partially addressed by Zhang et al. (2015), where user behavior such as skipping and viewing query completions is taken into account for query auto completion.

As the QAC process typically starts when a user enters the first character into the search box and ends until he clicks a completed query, a series of user interactions with

the QAC system involves examining the list of query completions, continuing to type, and clicking on a query in the list. Such complete interactions in the QAC process have not been well studied by previous QAC approaches. Li et al. (2014) adopt a click model to shed light on QAC user interactions, which consists of a horizontal component that captures the skipping behavior, a vertical part that depicts the vertical examination bias, and a relevance model that reflects the intrinsic relevance between a prefix and a query completion. This model is motivated by the fact that users frequently skip query completion lists even though such lists contain the final submitted query because the intended query is not ranked at top positions of the list.

Following (Li et al., 2014), Zhang et al. (2015) exploit implicit negative feedback in a QAC process, e.g., dwelling on query completion lists for a long time without selecting query completions ranked at the top. Such user feedback indicates that the user may not favor these unselected query completions although they are ranked at the top positions in the list. To model this negative feedback, the dwell time and the position of unselected query completions are taken into account. By incorporating a (static) relevance score with implicit negative feedback, the proposed model re-ranks the top-N queries initially returned by popularity. Finally, a case study verifies that the new model generally outperforms start-of-the-art QAC models for the cases that users submit new queries when reformulating older queries and that users have clear query intent so as to prefer disambiguated queries.

2.5 Summary

In this chapter, we have summarized related work on query auto completion, which is based on either probabilistic models or on a learning framework. In particular, the work on probabilistic models that we have surveyed so far is mainly based on either the search context (Bar-Yossef and Kraus, 2011; Cai et al., 2014b) or on user interactions (Li et al., 2015; Zhang et al., 2015) to estimate a probability of submitting a query completion. Despite their intuitiveness, these probabilistic QAC approaches have been shown to be outperformed by relatively simple methods such as the MPC model (Bar-Yossef and Kraus, 2011), regardless of their choice of how to compute the ranking scores for query completions. In addition, we have summarized learning-based query auto completion approaches that explore features from time-related characteristics and from user-specific characteristics. Such approaches use time series analysis to predict the future popularity of queries and on user behavior analysis by understanding the interaction data in a QAC engagement.

Finally, we have discussed some practical issues related to QAC, i.e., efficiency, presentation and interaction. Addressing these issues makes a QAC service fit together with search engines well. So far, researcher have paid less attention to these practical issues in QAC than to query completion ranking models even though QAC services have to run subject real-time constraints. Most processing steps in a QAC system can be done offline and the QAC ranking models can be trained regardless of the time consumption. Based on a trained QAC ranking model, the test phase only requires very limited time cost.

This chapter has introduced the task of query auto completion and its corresponding methods. In the chapters to come we build on the methods introduced here; the experi-

mental methodology and setup that we use to assess our proposals is introduced next.

3 Experimental Methodology

In this chapter, we move on to the experimental methodology for query auto completion (QAC) that is commonly used in later research chapters, Chapter 4–7. In particular, we present the experimental setup in §3.1, detail the benchmark datasets used for experiments in §3.2, and describe the metrics used for evaluations in §3.3.

3.1 Experimental Setup

In essence, QAC approaches deal with a *re-ranking* problem (Bar-Yossef and Kraus, 2011; Cai and de Rijke, 2016a; Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013). That is, as shown in Figure 3.1, QAC models focus on re-ranking a ranked list of query completions that is initially returned by a basic ranker when a user enters a prefix. In this flow, the initial query list, i.e., *query list 1* in Figure 3.1, is at least as long as the final re-ranked query list, i.e., *query list 2* in Figure 3.1. Thus, we can think of QAC models as *re-rankers*. The dominant types of signals used for re-ranking are either time-related or user-centered as described in Chapter 2. In addition, for evaluation purposes, in our

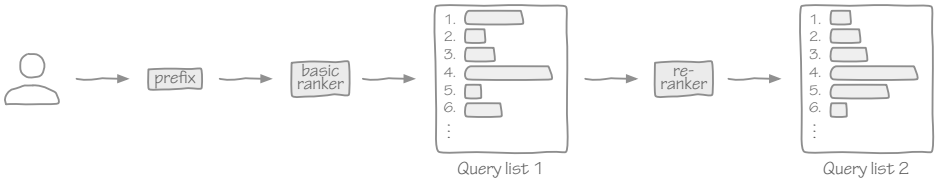


Figure 3.1: General flow of a QAC approach.

experiments, the typed prefixes are simulated from all possible prefixes of the submitted query. Typically, the prefix consists of 1 to 5 characters. This setting is consistently shared in later research chapters. Some very particular parameter settings will be discussed in the relevant research chapters.

For comparisons, note that we do not compare our approach with query suggestion methods, e.g., (Liao et al., 2011; Ma et al., 2008; Mei et al., 2008b), because such methods focus on the task of query suggestion, which is a subtly different task from QAC (Cai et al., 2014b). For instance, in the case of query suggestion, the user input typically is

a full *query* but it is only a *prefix* in QAC. Also, for generating the original query completion list to re-rank in our QAC task, only the candidates matching the typed prefix are taken into consideration. However, for a query suggestion task, each query in the query log could be included in the candidate list to re-rank given that it is deemed *relevant* to the user input. With this limitation we follow a standard practice in research on QAC (Bar-Yossef and Kraus, 2011; Cai et al., 2014b; Shokouhi and Radinsky, 2012; Whiting and Jose, 2014), where only appropriate QAC approaches are used as baselines rather than query suggestion approaches. In addition, QAC is different from query correction oriented tasks such as those discussed by Duan and Hsu (2011b), where the main focus is on correctly modifying a misspelled word or query.

3.2 Benchmark Datasets

For a QAC task, each record in a benchmark test collection has at least three main components: a submitted query, a user ID (or session ID) and a timestamp. The timestamp and the user ID (or session ID) are used for extracting time-sensitive and user-specific characteristics, respectively, while the query can be used to manually generate the query prefix and to count the appearance in the query logs. Besides, some additional information, e.g., the clicked URLs and their ranks, could be available too. Such data could be developed to infer user's search intent.

Three publicly available query log datasets, i.e., the AOL dataset (Pass et al., 2006), the MSN dataset (Craswell et al., 2009), and the Sogou dataset,[1] have been widely used for QAC training and evaluation. For instance, the AOL dataset is commonly used in time-sensitive QAC approaches (Cai and de Rijke, 2016a; Cai et al., 2014b; Whiting and Jose, 2014), as a relatively long period of query logs is provided, as well as in personalized methods (Bar-Yossef and Kraus, 2011; Shokouhi, 2013). The MSN dataset is preferred in (Cai and de Rijke, 2016a; Cai et al., 2016b) for personalizing query auto completion. And the Sogou dataset is used in (Whiting and Jose, 2014). Other query log datasets from modern commercial search engines, which record more search details but are not publicly available because of privacy or competitiveness issues, have also been applied for research (Jiang et al., 2014b; Li et al., 2015, 2014; Shokouhi and Radinsky, 2012; Zhang et al., 2015).

Table 3.1 shows an example of search session[2] in the well-known AOL query log dataset (Pass et al., 2006). In this case, this user submitted three queries in total. Each of the two previous queries, i.e., "*goodyear*" and "*hyundaiusa*", has one clicked URL, which is ranked at position 1 in the returned corresponding list of URLs. The last query, i.e., "*order hyundai parts*," has three clicked URLs, at position 2, 9 and 6, respectively. The AOL dataset is one of the few datasets that contain actual queries and is sufficiently large to guarantee statistical significance. The queries in the AOL dataset were sampled between March 1, 2006 and May 31, 2006. The MSN logs were recorded for one month in May 2006 from the MSN search engine. An early version of the Sogou dataset was released in 2008 but it was replaced by a bigger dataset in 2012. Privacy information has been removed as well as illegal query sessions. Statistics of the three publicly available

[1] `http://www.sogou.com/labs`

[2] Segmented by a fixed time window of 30 mins inactivity (Jones and Klinkner, 2008; Lucchese et al., 2011).

Table 3.1: Snapshot from a search session in the AOL query log dataset.

User ID	Query	Timestamp	Rank	URL
2722	goodyear	2006-05-21 09:10:29	1	http://www.goodyear.com
2722	hyundaiusa	2006-05-21 09:28:54	1	http://www.hyundaiusa.com
2722	order hyundai parts	2006-05-21 09:51:02	2	http://www.hypartswholesale.com
2722	order hyundai parts	2006-05-21 09:51:02	9	http://www.the-best-source.com
2722	order hyundai parts	2006-05-21 09:51:02	6	http://www.racepages.com

Table 3.2: Statistics of publicly available query log collections, i.e., the AOL dataset (AOL), the MSN dataset (MSN) and the Sogou dataset (Sogou), for query auto completion evaluation.

Statistic	AOL	MSN	Sogou
Language	English	English	Chinese
Start date	2006-03-01	2006-05-01	2008-06-01
End date	2006-05-31	2006-05-31	2008-06-30
# Days	92	31	30
# Records	36,389,567	14,921,285	43,545,444
# Queries	10,154,742	8,831,281	8,939,569
# Users	657,426	–	9,739,704
# Sessions	–	7,470,916	25,530,711
# URLs	1,632,788	4,975,897	15,095,269

query log collections presented in this section are provided in Table 3.2. Although these logs largely facilitate research on query auto completion in information retrieval, they are limited to the detailed user interaction with a QAC engine.

In addition, we use a dataset made available by the Netherlands Institute for Sound and Vision,[3] to which we will refer as "SnV" (Huurnink et al., 2010). SnV is one of the largest audiovisual archives in Europe. The SnV logs were recorded for one year between January 1, 2013 and December 31, 2013 using an in-house system tailored to the archive's online interface. We filter out a large volume of navigational queries containing URL substrings (.com, .net, .org, http, .edu, www.) from the datasets and remove queries starting with special characters such as &, $ and # from all datasets. The chapters in this thesis are mainly based on the AOL and MSN query log datasets. In particular, the AOL dataset is used in all research chapters, Chapter 4–7; the MSN dataset is used in Chapter 5 and 6; and the SnV dataset is used in Chapter 4.

3.3 Evaluation Measures

To evaluate the effectiveness of QAC rankings in all research chapters, Mean Reciprocal Rank (MRR) is a standard measure. For a prefix p of a query q in the query set Q, assume that a list of matching query completions $S(p)$ and the user's final submitted query q' are

[3] `http://www.beeldengeluid.nl`

given. Then, the Reciprocal Rank (RR) for this prefix is computed as follows:

$$RR = \begin{cases} \frac{1}{\text{rank of } q' \text{ in } S(p)}, & \text{if } q' \in S(p) \\ 0, & \text{otherwise.} \end{cases} \tag{3.1}$$

Finally, the MRR score is computed as the average of all RR scores for tested prefixes of queries in Q. From this formula, MRR can be taken as a precision-oriented metric.

The choice of MRR as a performance metric is common in settings that are concerned with finding a single known solution. However, because of the issues reported in (Hofmann et al., 2014) on the formulation of MRR (averaging over a non ratio-scale), a less problematic metric, i.e., *success rate at top K* (SR@K), is also used for QAC evaluation, which denotes the average ratio of the actual query that can be found in the top K query completion candidates over the test data. This metric is widely used for tasks whose ground truth consists of only one instance such as query completion (Jiang et al., 2014b).

For measuring the accuracy of predicting a query's future popularity in the time-sensitive QAC model (Chapter 4), Mean Absolute Error (MAE) is widely used:

$$MAE = \frac{1}{n} \sum_{i=1}^{n} |\hat{y}_i - y_i|, \tag{3.2}$$

where y_i is the true value and $\hat{y}_i$ is the prediction. MAE is an unbounded measure and is not strongly resilient to outliers. Therefore, its is often used along with another metric such as Symmetric Mean Absolute Percentage Error (SMAPE) to diagnose the forecast variation. SMAPE is defined as:

$$SMAPE = \frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{y}_i - y_i|}{\hat{y}_i + y_i}, \tag{3.3}$$

In contrast to MAE, SMAPE is bounded between 0 and 1. To evaluate the quality of QAC rankings, Mean Reciprocal Rank (MRR) is a standard measure.

For evaluating the D-QAC performance in Chapter 6, which is studied in (Cai et al., 2016b), the official evaluation metrics in the diversity task of the TREC 2013 Web track (Collins-Thompson et al., 2013), e.g., ERR-IA (Chapelle et al., 2009), α-nDCG (Clarke et al., 2008), NRBP (Clarke et al., 2009) and MAP-IA (Agrawal et al., 2009) can be borrowed. We take the metric α-nDCG at cutoff N as an example for evaluating the diversity of a QAC rankings, which extends the traditional nDCG metric (Järvelin and Kekäläinen, 2002) for aspect-specific rankings according to:

$$\alpha\text{-}nDCG@N = Z_N \sum_{i=1}^{N} \frac{\sum_{a \in A_p} g_{i|a} (1-\alpha)^{\sum_{j=1}^{i-1} g_{j|a}}}{\log_2(i+1)}, \tag{3.4}$$

where a is an aspect in the set of query aspects A_p, $g_{i|a}$ denotes the aspect-specific gain of the i-th query given aspect a, and the normalization constant Z_N is chosen so that the perfect QAC list gets an α-nDCG@N score of 1. The α-nDCG@N commonly is computed at a trade-off $\alpha = 0.5$ in order to give equal weights to both relevance and diversity and all rewards are discounted by a log-harmonic discount function of rank. Generally, these diversity metrics reward newly-added aspects and penalize redundant aspects of query completions. See the referred literature for details on how these metrics are computed.

In our experiments, the statistical significance of observed differences between the performance of two approaches is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}$ ($^{\blacktriangledown}$) for significant differences at level $\alpha = .01$ and $^{\triangle}$ ($^{\triangledown}$) at level $\alpha = .05$.

3.4 Summary

In this chapter, we have detailed the currently established methodology for evaluating query auto completion rankings. In particular, we have presented the general experimental setup for query auto completion in information retrieval and then described the publicly available benchmark query log collections used for query auto completion. In addition, we have introduced the most widely used metrics for QAC evaluation, including the measure for the accuracy of query completion as well as for the diversification of query auto completion.

So far, we have introduced the background and experimental methodology of this thesis. From the next chapter, we will focus on our investigation on query auto completion in information retrieval.

4

Prefix-adaptive and Time-sensitive Personalized Query Auto Completion

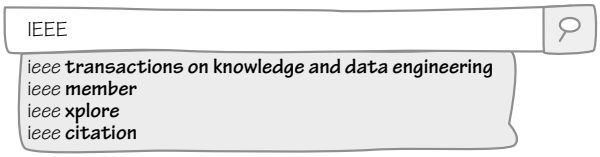
A common and effective approach in previous work on query auto completion (QAC) is to extract, for each prefix, all past queries that expand it from a period of query logs, and then rank them by their popularity (Bar-Yossef and Kraus, 2011; Shokouhi and Radinsky, 2012; Strizhevskaya et al., 2012), i.e., by the number of times they have been submitted. This assumes that the current and future query popularity are the same as the past query popularity. Although this kind of approach results in satisfactory QAC performance on average, it is far from optimal since it fails to take strong clues from time, trend and user-specific context into consideration while such information often influences the queries most likely to be typed.

For instance, as illustrated in Figure 4.1, personalized QAC may inject the most popular completions from a user as query completions for that particular user; see Figure 4.1a (not personalized) and Figure 4.1b (personalized). From Figure 4.2a and Figure 4.2b, obtained using Google Trends,[1] we see that query popularity strongly depends on time. We observe a clear burst for the query "*MH17*" around July 18, 2014. We also see that query popularity may be subject to cyclic phenomena: there is a yearly cycle for the query "*New year*" and a weekly cycle for the query "*Movie*." These phenomena can be explored to forecast a query's future popularity, which motivates a QAC approach that takes both the temporal aspect and the personal context into account. The work in this chapter is an attempt towards this objective.

In addition, some user inputs are easier to complete than others, depending on the "popularity" of the prefix, as measured by the number of returned query completions. Consider, for instance, Table 4.1, which displays a search session with three queries from the well-known AOL query log (Pass et al., 2006). For the sake of the example, let us assume that we have not yet seen the last query (row 4 in Table 4.1a, the query "*jsonline*"), and that there are three lists in Table 4.1b of query completions with the initial prefix "*js*," "*jso*" and "*json*," respectively, of this query "*jsonline*," for which we want to recommend completions. A regular baseline based on the most popular query is applied to produce these completions (Bar-Yossef and Kraus, 2011). More query completions are returned for the first prefix (row 2 in Table 4.1b) than for the second and third prefix (rows 3 and 4, respectively, in Table 4.1b). This means that, for a single popularity-based

[1] `http://www.google.com/trends`

(a) Completion of the prefix *"IEEE"* without logging in.



(b) Completion of the prefix *"IEEE"* after logging in.

Figure 4.1: Query auto completions of the typed prefix "*IEEE*" under different settings.

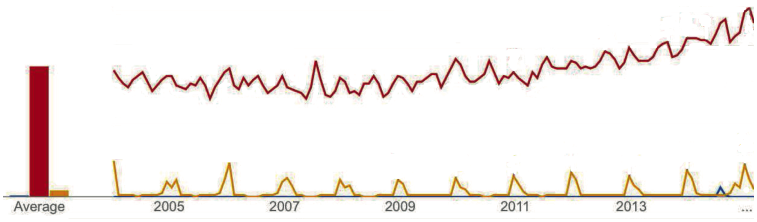
QAC approach, if the user inputs the prefix "*jso*" (or "*json*"), then, since there are few queries that start with "*jso*" (or "*json*"), the auto completion task is easier and is likely to produce better predictions. On the other hand, if the user's inputs are the letters "*js*" as a prefix, the large number of possible completions makes the prediction task harder. We treat the prefix "*jso*" (or "*json*") as a long-tail prefix and the prefix "*js*" as a normal one if we set the threshold $N = 10$ for the number of returned query completions. This rationale motivates us to work with a modified QAC model to adaptively deal with long-tail prefixes.

In this chapter, we focus on how to incorporate information of time-sensitivity with that of user-personality for boosting the performance of QAC. This chapter addresses research questions **RQ1**–**RQ6** as specified in §1.1. In particular, we propose to return the top N query completions by predicted popularity based on their cyclic behavior and recent trend, and then rerank these completions by user-specific context so as to output a final list of query completions. We propose two time-sensitive QAC models, λ-TS-QAC and λ^*-TS-QAC, that employ either a fixed trade-off λ or an optimal trade-off λ^* to answer the following research questions:

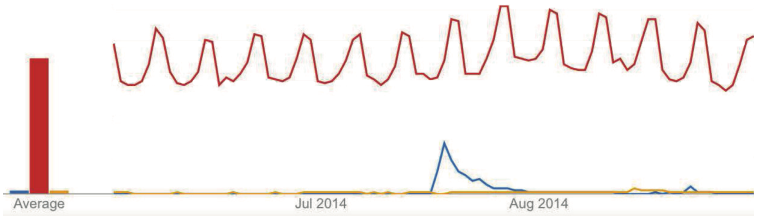
RQ1 As a sanity check, what is the accuracy of query popularity prediction generated by various models?

RQ2 How do our time-sensitive QAC models (λ-TS-QAC and λ^*-TS-QAC) compare against state-of-the-art time-sensitive QAC baselines?

We propose to predict a query's future popularity based on its recent trends as well as its cyclic popularity behavior, with a trade-off controlling these two parts when combining them together for a final ranking score. Such a trade-off can be assigned with a fixed value or can be individually optimized for each particular query based on a regression model. Our results show that λ^*-TS-QAC performs better in terms of MAE and SMAPE than all aggregation- and trend-based baselines, as well as λ-TS-QAC. We use the predicted query popularity to rank query completions for answering **RQ2** and find that λ^*-TS-QAC



(a) Relative query popularity in the past few years (2004–present).



(b) Relative query popularity within 3 months (June, 2014–August, 2014).

Figure 4.2: Relative query popularity for different queries over time. Queries: "*MH17*" in blue, "*Movie*" in red and "*New year*" in yellow. The snapshot was taken on Thursday, February 12, 2015.

outperforms the baseline as well as λ-TS-QAC in terms of MRR for most cases.

After incorporating the personal information from a user's search context into our time-sensitive QAC model, we propose a hybrid QAC model λ^*-H-QAC that considers both time-sensitivity and user personalization to compare with an n-gram based hybrid model λ^*-H$_G$-QAC so as to answer the research questions:

RQ3 Does λ^*-H-QAC outperform time-sensitive QAC methods, e.g., λ^*-TS-QAC?

RQ4 How does λ^*-H-QAC compare against personalized QAC method using n-gram based query similarity?

RQ5 How does λ^*-H-QAC compare against λ^*-H$_G$-QAC?

Finally, in this chapter, an extension of λ^*-H-QAC, named λ^*-H$'$-QAC, is proposed to deal with long-tail prefixes, i.e., unpopular prefixes, by optimizing the contributions from the predicted query popularity and from the user-specific context. To verify its effectiveness, we address another research question:

RQ6 How does λ^*-H$'$-QAC compare against λ^*-H-QAC on long-tail prefixes? And on all prefixes?

We prove that, after integrating the user-centered search context with our time-sensitive QAC model, our proposal, i.e., a prefix-adaptive hybrid QAC approach, further boosts the accuracy of ranking query completions.

Our contributions in this chapter can be summarized as follows:

(1) We address the challenge of query auto completion in a novel way by considering both time-sensitive query popularity and user-specific context.

Table 4.1: An AOL session example (Top) and lists of top ten (at most) candidates returned based on frequency after typing the corresponding prefix (Bottom).

(a) An AOL session example.

1	SessionID	UserID	Query	Time
2	310	1722	badjocks	20060523, 10:17:44
3	310	1722	jsonline	20060523, 10:21:30
4	310	1722	jsonline	20060523, 10:21:37

(b) Lists of top ten (at most) candidates after typing corresponding prefix.

1	Prefix	List of top ten (at most) QAC candidates
2	js	jsonline, jsu, js, jstor, js online, jsfirm, jsp, js collection, jso, jscfcu
3	jso	jsonline, jso
4	json	jsonline

(2) We propose a new query popularity prediction method that explores the cyclic behavior and recent trend of query popularity, achieving a better approximation to the real popularity in terms of Mean Average Error (MAE).
(3) We extend our hybrid QAC model to deal with long-tail prefixes by optimizing the contributions from query popularity and user-specific context.

The remainder of this chapter is organized as follows. Our prefix-adaptive and time-sensitive personalized query auto completion approach is described in §4.1, while the experimental setup is presented in §4.2; §4.3 reports our experimental results. We conclude in §4.4 where we also recommend several directions for future work.

4.1 Approach

In this section we describe our time-sensitive personalized query auto completion approach, a hybrid model that not only inherits the merits of time-sensitive query auto completion but also considers a user's personal context. Table 4.2 provides an overview of various QAC approaches; the baselines (rows 1–3) are described in the literature; we detail our models (rows 4–10) in three steps: time-sensitive QAC first, then personalized QAC, and, finally, hybrid QAC.

4.1.1 Periodicity and Trend Based QAC

We propose a time-sensitive QAC method, TS-QAC, that ranks query completions by predicted query popularity (i.e., their frequency) based on periodicity and recent trends so that we can detect both cyclicity and instantly frequent queries. TS-QAC not only inherits the merits of time-series analysis for long-term observations of query popularity, but also considers recent variations in query counts. Specifically, we predict a query q's next-day popularity $\tilde{y}_{t_0+1}(q, \lambda)$ at day $t_0 + 1$ after day t_0 by both its recent trend and

Table 4.2: Description of query popularity prediction methods and QAC approaches.

Approach	Description	Source
P_k	Predict a query's future popularity by aggregating the frequency in the past k days	(Shokouhi and Radinsky, 2012)
MPC-ALL	Rank query completions according to the popularity on the whole log	(Bar-Yossef and Kraus, 2011)
MPC-R-TW	Rank query completions according to the popularity within a fixed time window	(Whiting and Jose, 2014)
O-MPC-R	Rank query completions according to the popularity within an optimal time window	(Whiting and Jose, 2014)
P_{trend}	Predict a query's future popularity by its recent trend as described in (4.3)	This chapter
λ^*-TS-QAC	A TS-QAC model with an optimal parameter λ^* used in (4.1), achieved by (4.7)	This chapter
Personalized QAC	Rank query completions according to the similarity to previous queries as (4.8)	This chapter
λ-H-QAC	A hybrid model integrating λ-TS-QAC as (4.13) and personalized QAC as (4.14)	This chapter
λ^*-H-QAC	A hybrid model integrating λ^*-TS-QAC by (4.7) and personalized QAC as (4.14)	This chapter
G-QAC	A hybrid model integrating MPC-ALL and n-gram based personalization as (4.12)	This chapter
λ^*-H_G-QAC	A hybrid model integrating λ^*-TS-QAC and n-gram based personalization as (4.12)	This chapter
λ^*-H′-QAC	A modified λ^*-H-QAC model with long-tail prefix detection added	This chapter

periodicity with a free parameter λ ($0 \leq \lambda \leq 1$) controlling the contribution of each component:

$$\tilde{y}_{t_0+1}(q, \lambda) = \lambda \cdot \hat{y}_{t_0+1}(q)_{trend} + (1 - \lambda) \cdot \bar{y}_{t_0+1}(q)_{peri}, \tag{4.1}$$

where $\lambda = 1$ for aperiodic queries and $0 \leq \lambda < 1$ for periodic queries. The term $\hat{y}_{t_0+1}(q)_{trend}$ is estimated via a linear aggregation of predictions from N_{days} recent observations:

$$\hat{y}_{t_0+1}(q)_{trend} = \sum_{i=1}^{N_{days}} norm(\omega_i) \cdot \hat{y}_{t_0+1}(q, i)_{trend}, \tag{4.2}$$

where $norm(\omega_i)$ normalizes the contributions from each day to ensure $\sum_i \omega_i = 1$.

We introduce a temporal decay function to output the weight before normalizing as $\omega_i = f^{TD(i)-1}$, where f is a decay factor and $TD(i)$ refers to the interval from day i to the future day $t_0 + 1$. We identify the highest prediction accuracy parameter N_{days} for

each query based on past observations in the whole log using a multiple linear regression model, following (Whiting and Jose, 2014). The prediction $\hat{y}_{t_0+1}(q,i)_{trend}$ from each day i ($i = 1, \ldots, N_{days}$) is derived from the first order derivative of q's daily count $C(q,t)$ as:

$$\hat{y}_{t_0+1}(q,i)_{trend} y_{t_0-TD(i)}(q,i) + \int_{t_0-TD(i)}^{t_0+1} \frac{\partial C(q,t)}{\partial t} \mathrm{d}t, \tag{4.3}$$

where $y_{t_0-TD(i)}(q,i)$ is the observed query count of q at day i.

The periodicity term $\bar{y}_{t_0+1}(q)_{peri}$ in (4.1) is smoothed by simply averaging the recent M observations y_{t_p} at preceding time points $t_p = t_0 + 1 - 1 \cdot T_q, \ldots, t_0 + 1 - M \cdot T_q$ in the log:

$$\hat{y}_{t_0+1}(q)_{peri} = \tfrac{1}{M} \textstyle\sum_{m=1}^{M} y_{t_0+1-m \cdot T_q}(q), \tag{4.4}$$

where T_q denotes q's periodicity.

For detecting cyclic aspects of a query q's frequency, we use the autocorrelation coefficients (Chatfield, 2004), which measure the correlation between N_s successive count observations $C(q,t)$ at different times $t = 1, 2, \ldots, N_s$ in the query log. The correlation is computed between a time series and the same series lagged by i time units as:

$$r_i = \frac{\sum_{t=1}^{N_s-i} (C(q,t) - \bar{x}_1)(C(q,t+i) - \bar{x}_2)}{\left(\sum_{t=1}^{N_s-i} (C(q,t) - \bar{x}_1)^2\right)^{\frac{1}{2}} \left(\sum_{t=i+1}^{N_s} (C(q,t+i) - \bar{x}_2)^2\right)^{\frac{1}{2}}}, \tag{4.5}$$

where $\bar{x}_1$ is the mean of the first $N_s - i$ observations and $\bar{x}_2$ is the mean of the last $N_s - i$ observations. For N_s reasonably large, the denominator in (4.5) can be simplified by approximation. First, the difference between the means $\bar{x}_1$ and $\bar{x}_2$ can be ignored. Second, the difference between summations over observations 1 to $N_s - i$ and $i + 1$ to N_s can be ignored. Accordingly, r_i can be approximated by:

$$r_i \approx \frac{\sum_{t=1}^{N_s-i} (C(q,t) - \bar{x})(C(q,t+i) - \bar{x})}{\sum_{t=1}^{N_s} (C(q,t) - \bar{x})^2}, \tag{4.6}$$

where $\bar{x} = \sum_{t=1}^{N_s} C(q,t)$ is the overall mean.

In addition, we choose an optimal λ^* by minimizing the metric *Mean Absolute Error* (MAE) (described in §4.2) as:

$$\lambda^* = \arg \min_{0 \le \lambda \le 1} \frac{1}{|Q|} \cdot \frac{1}{|L_v|} \sum_{q \in Q} \sum_{s=1}^{|L_v|} |\tilde{y}_s(q,\lambda) - y_s(q)|, \tag{4.7}$$

where $\tilde{y}_s(q,\lambda)$ and $y_s(q)$ are the predicted and the true query counts at day s in the validation period (L_v days), respectively.

Algorithm 1 details the major steps of our time-sensitive QAC method. We write λ-TS-QAC for the version with a fixed λ and λ^*-TS-QAC for the version where an optimal λ^* is chosen. We fix the optimal number of days for predicting the popularity trend by minimizing the MAE in line 10 in Algorithm 1.

Algorithm 1 Time-sensitive query auto completion (TS-QAC).

Input: All queries: Q;
Length of training and validation days: L_t and L_v;
Querying time: t_0;
Number of returned completions: N;
Output: Predictions: $\bar{Q} = \{\bar{y}_{t_0+1}(q) : q \in Q\}$;
Top N completions of each prefix of all queries;

for each $q \in Q$ **do**
  $T_q \leftarrow autocor(Count(q))$;
  for $i = 1, \cdots, L_t$ **do**
    for $j = 1, \cdots, L_v$ **do**
      $\hat{y}_{t_0+1}(q)_{trend}[j] \leftarrow Regression(Count(q)[1:i])$;
      $AbsoluteError[j] \leftarrow \hat{y}_{t_0+1}(q)_{trend}[j] - y_{t_0+1}(q)|$;
    end for
    $MAE(i) \leftarrow mean(AbsoluteError)$;
  end for
  $N_{days} \leftarrow \arg\min_{1 \le i \le L_t} MAE(i)$;
  Update $\hat{y}_{t_0+1}(q)_{trend}$ with optimal N_{days} and Compute $\bar{y}_{t_0+1}(q)_{peri}$;
end for
Find an optimal λ^* by (4.7);
$\lambda \leftarrow \lambda^*$;
for each $q \in Q$ **do**
  $\tilde{y}_{t_0+1}(q, \lambda) \leftarrow \lambda \times \hat{y}_{t_0+1}(q)_{trend} + (1 - \lambda) \times \bar{y}_{t_0+1}(q)_{peri}$;
end for
for each $q \in Q$ **do**
  for each prefix p of q **do**
    Return top N completions of p ranked by $\tilde{y}_{t_0+1}(q, \lambda)$;
  end for
end for

4.1.2 Personalized QAC

Next, we extend our time-sensitive QAC model described in §4.1.1 with personalized QAC. After sorting the queries with typed prefix p by predicted popularity following (4.1), we are given a ranking list of top N query completions. Let $\mathcal{S}(p)$ represent the set of returned top N query completions of prefix p.

Our personalized QAC works by scoring candidates $q_c \in \mathcal{S}(p)$ using a combination of similarity scores $Score(Q_s, q_c)$ and $Score(Q_u, q_c)$, where Q_s relates to the recent queries in the current search session and Q_u refers to those of the same user issued before, if available, as:

$$Pscore(q_c) = \omega \cdot Score(Q_s, q_c) + (1 - \omega) \cdot Score(Q_u, q_c), \tag{4.8}$$

where ω controls the weight of the individual components. Personalized QAC works at the session-based and user-dependent level.

To compute the required similarity scores, we first consider how to represent queries

in Q_s and Q_u. A naive approach would be to represent a query by n-grams or its terms as "a bag of words." The resulting similarity measure can capture syntactic reformulations. However, the problem is that queries are short, and thus their vocabulary is too sparse to capture semantic relationships. In order to overcome this sparsity problem, we use another solution to measure similarity. We observe in our datasets (see Table 4.3 and Figure 4.4) that users often request the same query or reformulate the query by modifying previous ones within the same session. We treat a user's preceding queries Q_s in the current session and their preceding queries Q_u in the historical log as context to personalize QAC where we measure similarity at the character level.

We represent a query $q_s \in Q_s$ and $q_c \in S(p)$ by the terms as $\{w_{s1}, w_{s2}, \ldots, w_{sm}\}$ and $\{w_{c1}, w_{c1}, \ldots, w_{cn}\}$ and let $N(w_*, q_*)$ denote the count of w_* appearing in q_*. We estimate the similarity between q_c and Q_s as a conditional probability:

$$\begin{aligned} Score(Q_s, q_c) &= p(q_c \mid Q_s) \\ &= \sum_{q_s \in Q_s} norm(\omega_s) \cdot p(q_c \mid q_s), \end{aligned} \tag{4.9}$$

where $norm(\omega_s)$ introduces a decay function $\omega_s = f^{TD(s)-1}$ as in (4.2) except that here $TD(s)$ refers to the interval between q_c and q_s, and $p(q_c \mid q_s)$ is calculated following (Cai et al., 2014a) as:

$$\begin{aligned} p(q_c \mid q_s) &= \prod_{w_{ci} \in q_c} p(w_{ci} \mid q_s)^{N(w_{ci}, q_c)} \\ &= \prod_{w_{ci} \in q_c} p(w_{ci} \mid W(w_{ci}))^{N(w_{ci}, q_c)}, \end{aligned} \tag{4.10}$$

where $W(w_{ci}) = \{w : \ w \in q_s \mid w[0] = w_{ci}[0]\}$ is a set of terms in q_s sharing the same start with w_{ci}, and define

$$\begin{aligned} p(w_{ci} \mid W(w_{ci})) &\equiv Similarity(w_{ci}, W(w_{ci})) \\ &= \frac{1}{|W(w_{ci})|} \sum_{w_j \in W(w_{ci})} Similarity(w_{ci}, w_j) \\ &= \frac{1}{|W(w_{ci})|} \sum_{j=1}^{|W(w_{ci})|} \frac{len(common(w_{ci}, w_j))}{\min(len(w_{ci}), len(w_j))}, \end{aligned}$$

where $len(common(w_{ci}, w_j))$ is the maximal length of common string appearing in w_{ci} and w_j from the beginning.

We compute $Score(Q_u, q_c)$ in a different manner from $Score(Q_s, q_c)$ in (4.9) because in this setting it is desirable to consider both query count and time interval. We output $Score(Q_u, q_c)$ as:

$$Score(Q_u, q_c) = p(q_c \mid Q_u) = \sum_{q_u \in Q_u} norm(\omega_u) \cdot p(q_c \mid q_u), \tag{4.11}$$

where $norm(\omega_s)$ only depends on the query count—we assume that frequent queries reflect a user's personal search clues.

Algorithm 2 Hybrid QAC model

Input: Predictions: $\bar{Q}$;
user: u;
prefix: p;
number of query completions to be reranked: N;
fixed trade-off: γ;
Output: Ranked list of top N query completions of p;
1: Produce $S(p)$ consisting of top N query completions by (4.1);
2: List u's queries Q_u and Q_s;
3: **for** each $q_c \in S(p)$ **do**
4: Compute $TSscore(q_c)$ based on (4.13);
5: **for** each $q_s \in Q_s$ **do**
6: $p(q_c \mid q_s) = Similarity(q_c, q_s)$;
7: **end for**
8: Compute $Score(Q_s, q_c)$ based on (4.9);
9: **for** each $q_u \in Q_u$ **do**
10: $p(q_c \mid q_u) = Similarity(q_c, q_u)$;
11: **end for**
12: Compute $Score(Q_u, q_c)$ based on (4.11);
13: Compute $Pscore(q_c)$ based on (4.8) and (4.14);
14: **end for**
15: Re-rank $S(p)$ by $HQscore(q_c)$ based on (4.12);
16: **Return** a reranked list of $S(p)$;

4.1.3 Hybrid QAC

We introduce a hybrid QAC model that combines time-sensitive QAC (TS-QAC) with personalized QAC. First, TS-QAC produces a list of query completions $S(p)$ of prefix p. We assign $TSscore(q_c)$ to each candidate $q_c \in S(p)$ using its predicted popularity, i.e., $\tilde{y}_{t_0+1}(q_c, \lambda)$ in (4.1). Like (Bar-Yossef and Kraus, 2011), we then define our hybrid models as convex combinations of two scoring functions:

$$Hscore(q_c) = \gamma \cdot TSscore(q_c) + (1 - \gamma) \cdot Pscore(q_c). \tag{4.12}$$

As $TSscore(q_c)$ and $Pscore(q_c)$ use different units and scales, they need to be standardized before being combined. We standardize $TSscore(q_c)$ (used in (Bar-Yossef and Kraus, 2011)) as:

$$TSscore(q_c) \leftarrow \frac{\tilde{y}_{t_0+1}(q_c, \lambda) - \mu_T}{\sigma_T}, \tag{4.13}$$

where μ_T and σ_T are the mean and standard deviation of the predicted popularity of queries in $S(p)$. Similarly, we use (4.8) to obtain

$$Pscore(q_c) \leftarrow \frac{Pscore(q_c) - \mu_P}{\sigma_P}, \tag{4.14}$$

where μ_P and σ_P are the mean and standard deviation of similarity scores of queries in $S(p)$. Algorithm 2 describes our hybrid QAC model, which requires a ranked list of

Algorithm 3 Optimization on long-tail prefixes (OLP)

Input: Predictions: $\bar{Q}$;
prefix set: $\bar{P}$;
number of query completions to be reranked: N;
threshold: Num;
Output: Subset $\bar{P}_1$ for long-tail prefixes and $\bar{P}_2$ for others;

```
1: P̄1 = P̄2 = {};
2: for each prefix p ∈ P̄ do
3:    Produce S(p) consisting of top N query completions by (4.1);
4:    if N < Num then
5:       P̄1 = P̄1 ∪ {p};
6:    else
7:       P̄2 = P̄2 ∪ {p};
8:    end if
9:    for each q_c ∈ S(p) do
10:      Compute TSscore(q_c) based on (4.13);
11:      Compute Pscore(q_c) based on (4.8) and (4.14);
12:   end for
13: end for
14: Apply linear regression on P̄1, producing an optimal weight γ̄ in (4.12);
15: Return γ̄, P̄1 and P̄2;
```

query completions along with their predicted popularity as produced by Algorithm 1; (4.12) provides the overall ranking score (see Algorithm 2, row 15).

We write λ-H-QAC to refer to the hybrid combination of λ-TS-QAC (used at line 4 in Algorithm 2) and the personalization approach described in the previous section; we write λ^*-H-QAC for the variant where λ has been optimized according to (4.7). For comparison, we also introduce other combined QAC models that consider query popularity and personalization. For instance, the G-QAC model derives the $TSscore(q_c)$ score in (4.12) by MPC-ALL and $Pscore(q_c)$ by using an n-gram representation similarity. Similarly, λ^*-H$_G$-QAC model generates $TSscore(q_c)$ by our λ^*-TS-QAC (in §4.1.1) and $Pscore(q_c)$ by the same n-gram representation similarity.

4.1.4 Modified λ^*-H-QAC (λ^*-H′-QAC)

In (4.12), the λ^*-H-QAC model assigns a fixed weight γ to $TSscore(q_c)$ and $1-\gamma$ to $Pscore(q_c)$ when calculating the final ranking score for the candidate q_c. All prefixes are handled equally when we score the candidates associated with them. However, we observe that some typed prefixes are easier to complete than others. This depends on the prefix popularity discussed in the beginning of this chapter. This observation motivates an extended ranking model upon λ^*-H-QAC. Rather than using a fixed weight γ for all prefixes, we assign an optimal weight $\bar{\gamma}$ to long-tail prefixes after checking their prefix popularity.

More specifically, to derive $\bar{\gamma}$ we first partition the prefixes in the training data into

Algorithm 4 λ^*-H$'$-QAC

Input: Predictions: $\bar{Q}$;
prefix set: $\bar{P}$;
number of query completions to be reranked : N;
optimal trade-off: $\bar{\gamma}$;
Output: Ranking list of top N query completions for each $p \in \bar{P}$;
1: **for** each prefix $p \in \bar{P}$ **do**
2: List $S(p)$;
3: **if** $p \in \bar{P}_1$ **then**
4: Perform λ^*-H-QAC for p with $\bar{\gamma}$ instead of γ in (4.12);
5: **else**
6: Perform λ^*-H-QAC for p with a fixed γ in (4.12);
7: **end if**
8: **end for**
9: **Return** a reranked list of $S(p)$;

two buckets according to the number of query completions returned by TS-QAC: long-tail and the other, i.e., $\bar{P}_1$ and $\bar{P}_2$ in Algorithm 3; then we directly perform a linear regression on $\bar{P}_1$ to generate an optimal weight $\bar{\gamma}$ for long-tail prefixes. Finally, λ^*-H$'$-QAC coincides with λ^*-H-QAC for normal prefixes with a fixed weight γ in (4.12) but unlike λ^*-H-QAC, for long-tail prefixes it uses the optimal weight $\bar{\gamma}$. The details are described in Algorithm 3 and Algorithm 4. We visualize the main steps of our model λ^*-H$'$-QAC in Figure 4.3, where most steps are done offline, including generating $S(p)$ by λ^*-TS-QAC and finding the optimal $\bar{\gamma}$ for long-tail prefixes, etc.

4.2 Experiments

In this section, §4.2.1 lists interesting observations from the datasets for experiments; we detail our settings and parameters in §4.2.2.

4.2.1 Datasets and Baselines

We use two query logs[2] in our experiments: AOL (Pass et al., 2006) and one made available by the Netherlands Institute for Sound and Vision,[3] to which we will refer as "SnV" (Huurnink et al., 2010). For consistency, we partitioned each log into two parts: a training set consisting of 75% of the query log, and a test set consisting of the remaining 25%. Traditional k-fold cross-validation is not applicable to temporally ordered data since it would obviously mess up the order (Gama et al., 2014). Queries in the training set were submitted before May 8, 2006 in the AOL dataset and before October 1, 2013 in the

[2]Other popular query logs used in recent research, such as the MSN log (Craswell et al., 2009) and the Sogou log (`http://www.sogou.com/labs`) were not used. The former lacks user IDs and the latter is too small.

[3]`http://www.beeldengeluid.nl`

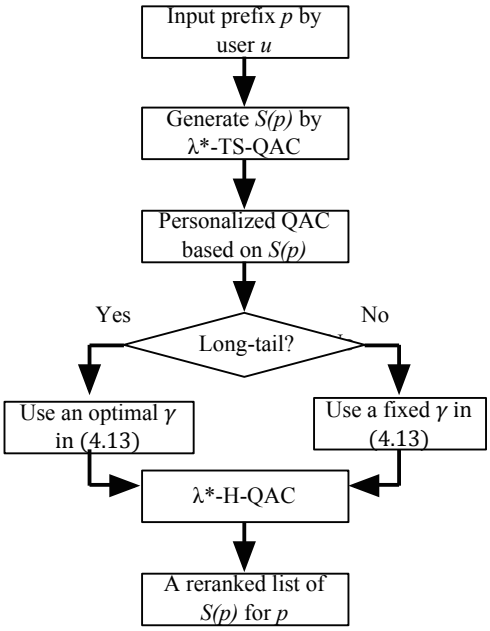


Figure 4.3: Main steps of the λ^*-H$'$-QAC model.

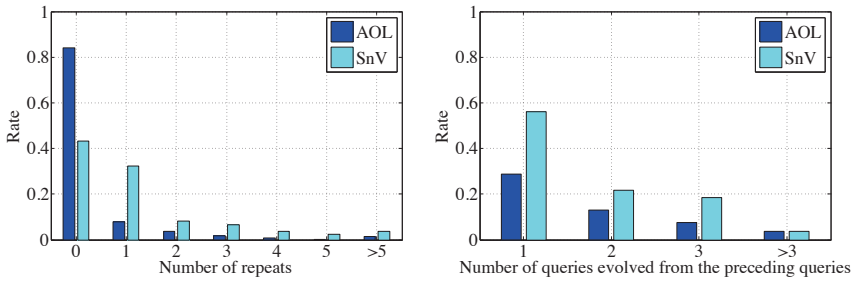
SnV dataset. We also use the last week of training data to generate optimal parameters: N_{days} in (4.2) and λ^* in (4.7).

In addition, only queries appearing in both partitions were kept. In total, 95,043 unique queries (21%) in the processed AOL and 6,023 (7%) in SnV show cyclic phenomena in terms of query frequency. Session boundaries are identified by 30 seconds of inactivity in the AOL dataset; in the SnV dataset a session boundary occurs when a query has no overlapping terms with the previous query as users routinely view audiovisual material during the search process; this can lead to periods of inactivity even though the user is still fully engaged in the search process (Huurnink et al., 2010). Table 4.3 details the statistics of the datasets.

We display the overlaps of queries with various ways of binning in Figure 4.4. Figure 4.4a shows the rates of unique $\langle$*user*, *query*$\rangle$ pairs posted at different numbers of repeats. A considerable number of queries are posted more than once by the same user within the training period (15.9% for AOL and 56.9% for SnV). The discrepancy between the rates can be explained by considering the type of user the search engine serves: general web users in the case of AOL vs. media professionals in the case of SnV. Figure 4.4b gives us the distribution of sessions containing queries that "evolved" from preceding queries within the session, where we say that query q_2 evolved from query q_1 if q_2 is issued after q_1 and shares at least one query term with q_1. Sessions with more than one query are considered. In total, there are 983,983 sessions in AOL and 35,942 in SnV left. Clearly, users reformulate a query very often from its previous queries. The difference between the sum of all rates (0.531 for AOL and 1 for SnV) is a consequence of different

Table 4.3: Statistics of the processed AOL and SnV datasets. Queries: Qs, Sessions: Ss, Users: Us.

	AOL		SnV	
Variables	Training	Testing	Training	Testing
#Qs	6,904,655	3,609,617	291,392	154,770
#Unique Qs	456,010	456,010	86,049	86,049
#Ss	5,091,706	2,201,990	176,893	102,496
#Unique Us	466,241	314,153	1051	804
Qs/Session	1.36	1.63	1.65	1.51
Qs/User	14.81	11.49	277.25	192.50



(a) Distribution of unique pair $\langle$*user*, *query*$\rangle$ at various number of repeats.

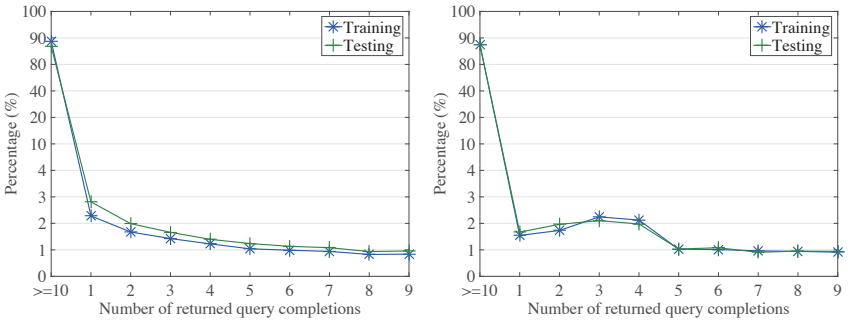
(b) Distribution of sessions containing various number of "evolved queries."

Figure 4.4: Query repeat rates (left) and variation rates (right) for AOL and SnV.

session segmentation methods.

In Figure 4.5, we plot the the ratios of long-tail prefixes among all prefixes in the training and test periods of the AOL and SnV datasets, respectively. For AOL, more than 13% of the prefixes are relatively rare in both training and test period. For SnV, fewer long-tail prefixes are detected, resulting in 12.5% and 12.7% for the training and test datasets, respectively. This means that, on average, we will encounter at least one long-tail prefix among every 10 prefixes. This finding motivates us to handle long-tail prefixes in a dedicated manner. In addition, we find that long-tail prefixes are often observed in search sessions where users resubmit queries that have been issued before in the current session, as shown in Figure 4.1. Interestingly, for both datasets the percentage of prefixes with few candidates (e.g., 1 and 2) is higher than those with more candidates (e.g., 8 and 9).

We consider several QAC baselines: (1) the most popular completion (MPC) QAC method based on the whole log, referred as MPC-ALL (Bar-Yossef and Kraus, 2011); (2) an MPC-based QAC method within recent time windows (TW = 2, 4, 7, 14 and 28 days, respectively) denoted as MPC-R-TW (Whiting and Jose, 2014); (3) a recent QAC method with an optimal time window referred as O-MPC-R, which learns the optimal time window for each prefix and performs best on the AOL data in (Whiting and Jose, 2014). To select the best baseline against which we compare our newly introduced



(a) AOL **(b) SnV**

Figure 4.5: Distribution of prefixes with varying numbers of returned query completions in the AOL and SnV datasets, tested on the training and test periods, respectively.

models, we compare the performance of the three approaches just listed and report the results in Table 4.4. For both datasets, O-MPC-R outperforms the other two approaches at different prefix lengths. For instance, it results in near 10% MRR improvements over MPC-ALL and MPC-R, respectively. Hence, we select O-MPC-R as the baseline for comparisons against our proposed models in latter experiments.

4.2.2 Settings

Following (Bennett et al., 2012), we set the factor $f = 0.95$ in the decay function in §4.1.1. For time-sensitive prediction, we use a fixed $\lambda = 0.5$ in (4.1) to compare with the results produced with an optimal λ^* returned by (4.7). To detect periodicity, we count queries per hour for AOL and per day for SnV because of the difference in time spans of the collected data. This means that for SnV, we compute $\hat{y}_{t_0+1}(q)_{peri}$ in (4.4) directly by averaging the day-level predictions $y_{t_0+1-m\cdot T_q}$, while for AOL, we firstly generate predictions per hour and then accumulate them to produce $y_{t_0+1-m\cdot T_q}$. For identifying trends, we use per day counts to overcome sparsity. For smoothing in (4.4), we set $M = 3$, as it performs best when M changes from 1 to 10 in our trials. In our time-sensitive QAC experiments, we are given a list of top N query completions; we set $N = 10$ as this is commonly used by many web search engines.

We balance the contributions of Q_s and Q_u in (4.8), if available, by setting $\omega = 0.5$, and construct Q_u using the ten most frequent queries of the user while collecting all preceding queries in the current session to form Q_s (see Table 4.3). In particular, for users without long-term search history, i.e., for cold-start users, we only consider their short-term search history in the current session for personalization. It could help if we use the long-term search logs from similar users seen in the training period. For instance, based on the preceding queries within a current session issued by a new user, we can find a group of seen users in the training period who have often submitted the same queries before. By using the long-term search logs of users in this group, we can model the interests of the new user for personalization. For personalized QAC comparisons, we

Table 4.4: Selecting our baseline. The performance of various baselines in terms of MRR, tested on the AOL and SnV datasets after typing 1 to 5 characters as prefix. The best performing baseline in each column is highlighted in boldface.

	AOL				
Model	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$
MPC-ALL	0.1090	0.1903	0.3018	0.3996	0.4813
MPC-R- 2 days	0.1093	0.1866	0.2989	0.3970	0.4681
MPC-R- 4 days	0.1082	0.1814	0.2902	0.3875	0.4593
MPC-R- 7 days	0.1120	0.1938	0.3107	0.4113	0.4830
MPC-R- 14 days	0.1140	0.1994	0.3217	0.4254	0.4985
MPC-R- 28 days	0.1147	0.2009	0.3233	0.4276	0.5076
O-MPC-R	**0.1175**	**0.2027**	**0.3267**	**0.4318**	**0.5087**
	SnV				
Model	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$
MPC-ALL	0.1573	0.2497	0.3281	0.4762	0.5438
MPC-R- 2 days	0.2467	0.3526	0.4917	0.6096	0.6913
MPC-R- 4 days	0.2281	0.3349	0.4751	0.5794	0.6681
MPC-R- 7 days	0.2209	0.3158	0.4519	0.5528	0.6327
MPC-R- 14 days	0.1953	0.2946	0.4318	0.5317	0.6108
MPC-R- 28 days	0.1731	0.2690	0.3873	0.5167	0.5731
O-MPC-R	**0.2519**	**0.3607**	**0.5034**	**0.6133**	**0.6992**

set the size of n-grams to $n = 4$, which has been recommended in string search (Litwin et al., 2007) to represent queries. For our hybrid models, we set $\gamma = 0.5$ in (4.12), which is also used by λ^*-H′-QAC for non-long-tail prefixes. In addition, we set the threshold $Num = 10$ when classifying prefixes in Algorithm 3.

4.3 Results and Discussion

In §4.3.1, we examine the performance of our time-sensitive QAC model in terms of its query popularity prediction performance, which we follow with a section about the trade-off of the parameter λ in §4.3.2. We examine the performance of various TS-QAC approaches in §4.3.3. Then, §4.3.4 details the effectiveness of our hybrid QAC model; §4.3.5 provides an analysis of the hybrid QAC model with various personalized QAC scenarios; §4.3.6 zooms in on the effect on QAC ranking by varying the contribution weight in hybrid QAC model. §4.3.7 compares the performance of combined QAC models. Finally, §4.3.8 and §4.3.9 detail the results of our model on long-tail prefixes.

Table 4.5: The forecast metrics produced by different methods on the AOL and SnV dataset. The best performer in each column is highlighted in boldface and the best performing baseline is underlined. Statistical significance of pairwise differences (λ-TS-QAC vs. the best baseline P_* and λ^*-TS-QAC vs. the best baseline P_*) are indicated.

	AOL		SnV	
Method	MAE	SMAPE	MAE	SMAPE
P_1	0.2906	<u>0.2278</u>	<u>1.2287</u>	<u>0.3104</u>
P_3	0.2944	0.2363	1.3739	0.3265
P_6	<u>0.2893</u>	0.2325	1.5751	0.3412
P_{trend}	0.2996	0.2313	1.2492	0.3117
λ-TS-QAC	$0.2848^{\triangle}$	$0.2197^{\blacktriangle}$	1.2291	$0.2959^{\blacktriangle}$
λ^*-TS-QAC	$\mathbf{0.2832}^{\triangle}$	$\mathbf{0.2145}^{\blacktriangle}$	$\mathbf{1.2067}^{\blacktriangle}$	$\mathbf{0.2813}^{\blacktriangle}$

4.3.1 Query Popularity Prediction Evaluation

Since the true popularity of query completions is unavailable at runtime, QAC ranking models sort query candidates according to their previously observed popularity (Bar-Yossef and Kraus, 2011) or predicted popularity inferred from previous logs (Shokouhi and Radinsky, 2012). In this section, we first evaluate the prediction accuracy on query popularity, and then measure the impact of these predictions on the quality of QAC rankings in §4.3.3.

Our time-sensitive prediction method considers both the recent and long-term query frequency to predict the popularity for future. To compare, the predicted query frequencies are aggregated over a past query log (used in (Shokouhi and Radinsky, 2012)) or only contributed over recent trend as described in (4.3). We denote the former by P_k where k is the number of previous days used for averaging ($k \in \{1, 3, 6\}$) and refer to the latter as P_{trend}. We do not take the prediction produced only by periodicity as baseline because of the unavailability of sufficiently many periodic queries (21% in AOL and 7% in SnV, see §4.2.1). Table 4.5 includes the forecast error rates of different methods on datasets. The numbers show that λ^*-TS-QAC performs better in terms of MAE and SMAPE than all aggregation- and trend-based baselines, as well as λ-TS-QAC.

Still focusing on Table 4.5, we take a closer look at the error rates produced by various models. The MAE achieved on AOL is much smaller than 1 due to the sparseness of query frequencies. Among the aggregated baselines, MAE favors P_6 and SMAPE prefers P_1 on AOL. However, for SnV, P_1 wins the competition on both metrics. The numbers show that with the exception of P_1 on SnV, our predictions are better than all aggregated baselines on both metrics. The differences are statistically significant on SMAPE but not so according to MAE. Overall, the competitive performance on the AOL dataset can be explained by the fact that compared to the daily query frequency used in the SnV dataset, the data here is less sparse and has lower variance.

Table 4.6: Performance in terms of MRR at prefix length $\#p$ ranging from 1 to 5 characters on the AOL and SnV datasets. The best performer in each column is highlighted in boldface. Statistically significant differences are determined against the baseline, i.e., O-MPC-R in Table 4.4, and marked in the upper right hand corner of the corresponding scores; statistically significant differences between λ^*-H-QAC and λ^*-H′-QAC vs. λ^*-TS-QAC are also detected and marked in the upper left hand corner of the corresponding scores.

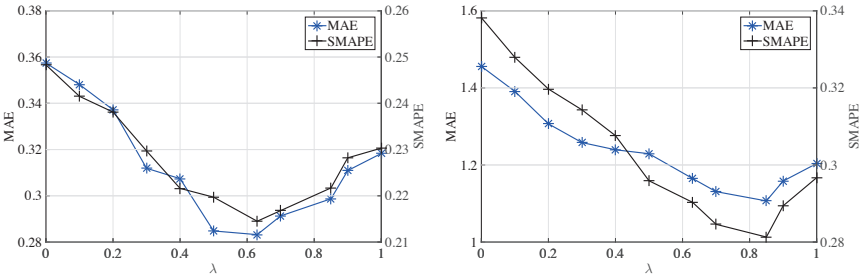
	AOL				
Model	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$
Baseline	0.1175	0.2027	0.3267	0.4318	0.5087
λ-TS-QAC	0.1169	$0.1982^{\triangledown}$	0.3270	$0.4390^{\triangle}$	$0.5115^{\triangle}$
λ^*-TS-QAC	0.1208	$0.2056^{\triangle}$	$0.3317^{\triangle}$	$0.4455^{\blacktriangle}$	$0.5143^{\blacktriangle}$
λ^*-H-QAC	**0.1224**$^{\blacktriangle}$	$0.2091^{\blacktriangle}$	$^{\triangle}0.3387^{\blacktriangle}$	$^{\triangle}0.4562^{\blacktriangle}$	$^{\triangle}0.5236^{\blacktriangle}$
λ^*-H′-QAC	**0.1224**$^{\blacktriangle}$	**0.2103**$^{\blacktriangle}$	$^{\blacktriangle}$**0.3408**$^{\blacktriangle}$	$^{\blacktriangle}$**0.4594**$^{\blacktriangle}$	$^{\triangle}$**0.5278**$^{\blacktriangle}$
	SnV				
Model	$\#p = 1$	$\#p = 2$	$\#p = 3$	$\#p = 4$	$\#p = 5$
Baseline	0.2519	0.3607	0.5034	0.6133	0.6992
λ-TS-QAC	0.2536	$0.3726^{\blacktriangle}$	$0.5117^{\triangle}$	$0.6296^{\blacktriangle}$	$0.7103^{\blacktriangle}$
λ^*-TS-QAC	$0.2637^{\blacktriangle}$	$0.3864^{\blacktriangle}$	$0.5193^{\blacktriangle}$	$0.6439^{\blacktriangle}$	$0.7203^{\blacktriangle}$
λ^*-H-QAC	**0.2662**$^{\blacktriangle}$	$0.3907^{\blacktriangle}$	$^{\blacktriangle}0.5355^{\blacktriangle}$	$^{\blacktriangle}0.6690^{\blacktriangle}$	$^{\blacktriangle}0.7491^{\blacktriangle}$
λ^*-H′-QAC	**0.2662**$^{\blacktriangle}$	**0.3913**$^{\blacktriangle}$	$^{\blacktriangle}$**0.5376**$^{\blacktriangle}$	$^{\blacktriangle}$**0.6702**$^{\blacktriangle}$	$^{\blacktriangle}$**0.7505**$^{\blacktriangle}$

4.3.2 Impact of the Trade-off Parameter λ

Next, we manually vary the parameter λ in (4.1) to determine the best prediction accuracy, with 0.01 increments. We show the results in Figure 4.6. For AOL (Figure 4.6a), λ^*-TS-QAC performs best in terms of prediction accuracy with $\lambda^* = 0.62$, suggesting a light emphasis on recent variations. We repeat our analysis on SnV and summarize the results in Figure 4.6b. The results are consistent with the overall AOL numbers. SnV receives an optimal value of $\lambda^* = 0.83$ in our experiments. This is due to the fact SnV contains fewer periodic queries than AOL and hence it favors predictions from the trend.

Another interesting finding on both datasets from Figure 4.6 is that MAE and SMAPE favor a relatively larger value of λ and they their behaviors are similar: MAE decreases as SMAPE comes down. This implies that (1) recent trends are important to predict future popularity; and (2) periodic phenomena also contribute as the errors go up if their contribution is removed (that is, with $\lambda = 1$).

Next, we have a close look at the optimal λ, i.e., λ^*. We find that for periodic queries, the optimal λ^* is often larger than 0.5. For instance, the mean of all these optimal λ^* in the AOL dataset is close to 0.6. In other words, the contribution from the cyclic behavior of query popularity is relatively less important than that from the recent trend. However, neither of them is negligible for predictions of query popularity as both weights are substantially larger than 0.



(a) Prediction accuracy at various λ on AOL.

(b) Prediction accuracy at various λ on SnV.

Figure 4.6: Impact of the trade-off parameter λ in TS-QAC on the accuracy of query popularity prediction for AOL and SnV.

4.3.3 Performance of TS-QAC Ranking

For **RQ2**, we use MPC-based models to generate QAC rankings for each prefix to compare with our results produced by time-sensitive QAC models, namely, λ-TS-QAC and λ^*-TS-QAC. Table 4.6 contains the evaluation results of different QAC models in terms of MRR. On both two datasets, each prefix is used to generate ten QAC rankings. For now, ignore the λ^*-H-QAC row as we will get to it later. All pairwise differences are detected and marked if statistically significant.

We find that λ^*-TS-QAC outperforms the baseline as well as λ-TS-QAC in terms of MRR, while λ-TS-QAC loses against the baseline for $\#p = 1$ and 2 on AOL. Specifically, λ^*-TS-QAC offers a maximal MRR increase against the baseline of 3.2% for $\#p = 4$, which is significant, and λ-TS-QAC brings an increase by up to 1.7% over the baseline for $\#p = 4$ on the AOL corpus. On the SnV dataset, we see the biggest performance improvements over the baseline: almost 7.1% for λ^*-TS-QAC and 3.3% for λ-TS-QAC, both when expanding a 2-character prefix. The limited improvement of λ-TS-QAC is probably due to predictions on occasional queries such as news search, whereas λ^*-TS-QAC smoothes it with cyclic phenomena for QAC.

With an optimal λ^* specifying the contributions from the recent trends and the cyclic behavior, the TS-QAC model produces better QAC rankings than with a fixed λ, as we can see by comparing λ-TS-QAC and λ^*-TS-QAC in Table 4.6. Generally, at different prefix lengths, λ^*-TS-QAC receives larger MRR gains over λ-TS-QAC on SnV than AOL. We attribute this to the volume of periodic queries in different datasets as discussed in §4.3.2.

4.3.4 Hybrid QAC Ranking Performance

Research question **RQ3** is aimed at examining whether a user's personal query similarity helps generate better QAC rankings. We first give the absolute MRR scores of λ^*-H-QAC in Table 4.6. For convenience, we report the MRR changes produced by comparing O-MPC-R against λ^*-H-QAC and λ^*-TS-QAC against λ^*-H-QAC in Table 4.7. With the appropriate regression model and query similarity measure, λ^*-H-QAC is able to

Table 4.7: MRR changes observed by comparing O-MPC-R against λ^*-H-QAC and λ^*-TS-QAC against λ^*-H-QAC, respectively, with a query prefix p of 1–5 characters on AOL and SnV query logs. The symbol "–" before MRR changes means λ^*-H-QAC outperforms the corresponding method. Statistical significance of pairwise differences (O-MPC-R vs. λ^*-H-QAC and λ^*-TS-QAC vs. λ^*-H-QAC) is indicated.

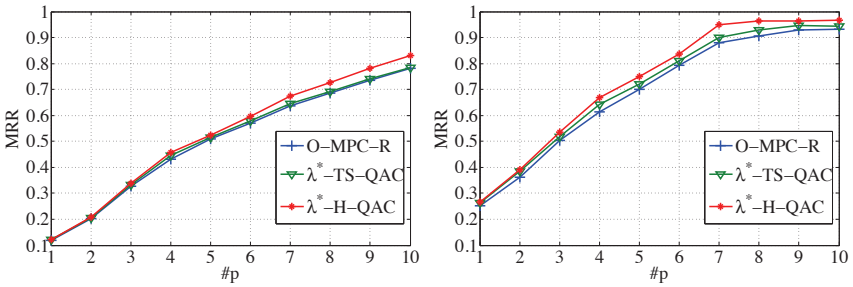
	AOL		SnV	
$\#p$	O-MPC-R	λ^*-TS-QAC	O-MPC-R	λ^*-TS-QAC
1	–4.00%$^{\blacktriangledown}$	–1.31%	–5.37%$^{\blacktriangledown}$	–0.94%
2	–3.06%$^{\blacktriangledown}$	–1.67%	–7.68%$^{\blacktriangledown}$	–1.10%
3	–3.54%$^{\blacktriangledown}$	–2.07%$^{\triangledown}$	–5.99%$^{\blacktriangledown}$	–3.03%$^{\blacktriangledown}$
4	–5.35%$^{\blacktriangledown}$	–2.35%$^{\triangledown}$	–8.33%$^{\blacktriangledown}$	–3.75%$^{\blacktriangledown}$
5	–2.85%$^{\blacktriangledown}$	–1.79%$^{\triangledown}$	–6.67%$^{\blacktriangledown}$	–3.84%$^{\blacktriangledown}$

marginally outperform the baselines on both query logs at each prefix length. However, despite the additional overhead of scoring similarity between queries, λ^*-H-QAC presents relatively small ($\sim$2%) improvements over λ^*-TS-QAC on AOL. This is due to the fact that no strongly differential features are explored yet for users.

Compared to AOL, λ^*-H-QAC on SnV achieves more relative MRR gains over the baselines and the MRR differences are generally enlarged for longer prefixes. In part, this may be due to the following. Firstly, AOL contains more queries than SnV, although these are spread sparsely over a three-month period. This could suggest that a search engine serving more queries is able to generate better completion candidates since it has a larger sample of similar behavior. Secondly, AOL is a more general search log across topics while SnV focuses on multimedia search. Thirdly, there may be underlying demographic differences between users of the two search logs that lead to changes in query distributions, for example, AOL covers more public users while SnV mostly serves for media professionals. Additionally, the higher performance of SnV as compared to AOL could be a consequence of the difference in user activity as *Qs/Us* in Table 4.3 indicates SnV users submit $\backsim$20 times more queries than AOL users.

Clearly, for both query logs, λ^*-H-QAC is considerably more effective with a longer prefix, see Table 4.6 and 4.7. To verify this, we examine the MRR scores with a longer prefix of up to 10 characters in Figure 4.7. We find that effectiveness converges more quickly on SnV than AOL when the length of prefix increases, probably because QAC is constrained by how much evidence is available, and a slightly longer prefix hugely narrows the number of possible completion candidates, certainly on the SnV dataset.

To illustrate the effectiveness of our model, we consider a test example from the AOL query log; see Table 4.8. Assume that our user has entered the prefix *vo* of the last query in this session, so that we need to recommend query completions for this prefix. The results shown in Table 4.9 are generated by the O-MPC-R, λ^*-TS-QAC and λ^*-H-QAC approaches, respectively. Clearly, the queries "volkswagon" and "volkswagen" benefit more from the search context than others as they are ranked at the top by the λ^*-H-QAC approach; this is because they are closely related to the earlier query "volks wagon" (line 4 in Table 4.8). Based on this insight, the queries "volkswagon" and "volkswagen" are more sensible completions. We could also explain it by introducing the semantic



(a) AOL **(b) SnV**

Figure 4.7: QAC performance in terms of MRR observed for each approach, with a query prefix p of 1–10 characters for the AOL and SnV query logs.

Table 4.8: An AOL test session.

1	SesionID	UserID	Query	Time
2	221	1038	euro car	20060519, 15:19:12
3	221	1038	eurocar	20060519, 15:19:53
4	221	1038	volks wagon	20060519, 15:21:07
5	221	1038	volkwagon	20060519, 15:21:21

similarity between the query completion and the previous queries in session. We consider the query pair likelihood ratio (Jones et al., 2006, (LLR)) in (4.15) as

$$LLR(q_1, q_2) = -2 \log \frac{L(q_2 \mid \neg q_1)}{L(q_2 \mid q_1)},$$

where $L(q_2 \mid \neg q_1)$ denotes the number of queries containing q_2 but without q_1, and $L(q_2 \mid q_1)$ indicates the volume of queries containing both q_1 and q_2, to see whether their co-occurrence in a search session is statistically significant. We find that, for instance, the pairs of query "volks wagon" and "volkswagon" or of query "volks wagon" and "volkswagen" co-occur relatively more often in sessions than other pairs.

4.3.5 Personalized QAC Performance Analysis

To help us answer **RQ4**, we compare the performance of λ^*-H-QAC with two personalized QAC scenarios (G-QAC and Personalized QAC listed in Table 4.2) and record the MRR scores of these two methods in Table 4.10. We also report the MRR changes produced by comparing G-QAC against λ^*-H-QAC, as well as Personalized QAC against λ^*-H-QAC in brackets in Table 4.10.

λ^*-H-QAC significantly outperforms G-QAC and Personalized QAC on both AOL and SnV in terms of MRR scores at all cases, which again confirms the above observations in Table 4.6. For AOL, Personalized QAC does not work well and its MRR scores are always substantially lower than those of G-QAC, suggesting that ranking query completions only according to query similarity on bigger dataset is not reliable because the

Table 4.9: Ranked lists of query completions for the prefix "*vo*".

	(a) Ranked by O-MPC-R.		(b) Ranked by λ^*TS-QAC.		(c) Ranked by λ^*H-QAC.
1	vonage	1	vonage	1	volkswagon
2	voyeur	2	voyeur	2	volkswagen
3	volvo	3	volvo	3	volvo
4	voyeurweb	4	volkswagon	4	volume
5	volcanoes	5	voyeurweb	5	volcanoes
6	volkswagon	6	volkswagen	6	volcano
7	volkswagen	7	volcanoes	7	vonage
8	voyeur web	8	volume	8	voyeur
9	volume	9	voyeur web	9	voyeurweb
10	volcano	10	volcano	10	voyeur web

number of query completions is very large and users often issue new queries. Interestingly, Personalized QAC outperforms G-QAC on SnV. We believe this can be attributed to (i) SnV users frequently issue similar queries in the current search session or in a long-term period, yielding distinguishable similarity scores for query completions; and (ii) the average number of queries of SnV users is larger, resulting in a better estimation of Q_u in (4.8).

Additionally, the MRR improvements of λ^*-H-QAC over G-QAC are still very high, indicating that MPC-ALL in G-QAC may often eliminate useful query completions. This negative effect is strengthened by low volumes of queries as the relative changes on SnV (around 15%) are larger than those on AOL (around 7%). We conclude that a small dataset suffers more from uncertainty on query popularity for ranking query completions.

4.3.6 Effect of the Contribution Weight γ

Next, we examine the effect on the overall QAC performance of varying the contribution weight γ in (4.12) in our hybrid QAC model, λ^*-H-QAC, from 0 to 1, on AOL and SnV. See Figure 4.8. For AOL (Figure 4.8a), if the value of γ used in λ^*-H-QAC goes up from 0 to 0.4, the performance increases more dramatically compared with the results under other settings ($0.4 < \gamma \leq 1$). If we rank query completions only by query similarity, i.e., $\gamma = 0$, the performance is worse than any other result. The MRR value of λ^*-H-QAC reaches its peak around $\gamma = 0.7$ for all cases, which shows that λ^*-H-QAC favors time-sensitive popularity over user's query similarity on AOL. This finding is confirmed when we average MRR values produced under different settings: $0 \leq \gamma \leq 0.5$ and $0.5 \leq \gamma \leq 1$ for each length of prefix. Obviously, the average MRR of the latter ($0.5 \leq \gamma \leq 1$) is higher for all cases.

In contrast to AOL, the optimal value of γ on SnV (Figure 4.8b) is around 0.3, which indicates that QAC ranking on SnV favors user's query similarity a bit more. The discrepancy between the optimal value of γ on SnV and the optimal value of γ on AOL can be explained by considering the number of issued queries of each user. Having sufficiently many personal queries results in effective personalized QAC on SnV. The MRR of SnV tends to be more sensitive to γ than that of AOL as it varies dramatically with

Table 4.10: MRR scores of G-QAC and Personalized QAC (Per. QAC), as well as MRR changes in bracket produced by comparing G-QAC against λ^*-H-QAC (in Table 4.6), and Personalized QAC against λ^*-H-QAC, respectively, with a query prefix p length of 1–5 characters tested on AOL and SnV query logs. Significant differences against λ^*-H-QAC are indicated.

	AOL		SnV	
$\#p$	G-QAC	Per. QAC	G-QAC	Per. QAC
1	0.1132$^{\blacktriangledown}$ (-7.52%)	0.0174$^{\blacktriangledown}$ (-85.78%)	0.2313$^{\blacktriangledown}$ (-13.11%)	0.2427$^{\blacktriangledown}$ (-8.83%)
2	0.1987$^{\blacktriangledown}$ (-4.97%)	0.0688$^{\blacktriangledown}$ (-67.10%)	0.3443$^{\blacktriangledown}$ (-11.88%)	0.3619$^{\blacktriangledown}$ (-7.35%)
3	0.3175$^{\blacktriangledown}$ (-6.26%)	0.1371$^{\blacktriangledown}$ (-59.52%)	0.4564$^{\blacktriangledown}$ (-14.76%)	0.5018$^{\blacktriangledown}$ (-6.29%)
4	0.4180$^{\blacktriangledown}$ (-8.37%)	0.2256$^{\blacktriangledown}$ (-50.55%)	0.5658$^{\blacktriangledown}$ (-15.43%)	0.6197$^{\blacktriangledown}$ (-7.37%)
5	0.4981$^{\blacktriangledown}$ (-4.88%)	0.3312$^{\blacktriangledown}$ (-36.75%)	0.6353$^{\blacktriangledown}$ (-15.19%)	0.7098$^{\blacktriangledown}$ (-5.25%)

the increase of γ, especially when $0.5 \leq \gamma \leq 1$. The overall MRR score of λ^*-H-QAC is better than that produced by just setting $\gamma = 0$ or $\gamma = 1$, which is consistent with our findings for AOL.

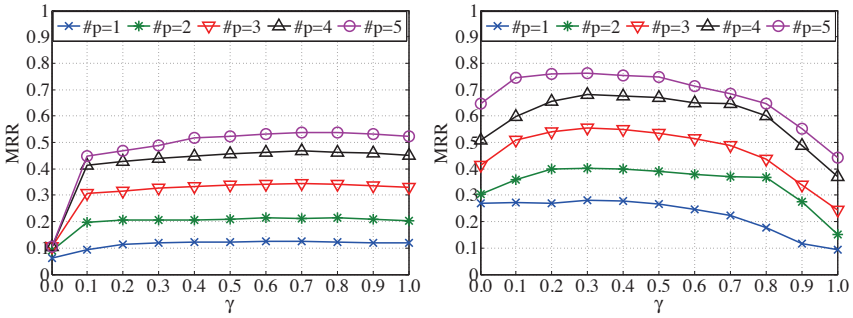
4.3.7 Performance of Combined QAC Models

To answer **RQ5**, we compare λ^*-H$_G$-QAC (in Table 4.2) with λ^*-H-QAC (MRR scores reported in Table 4.6). The MRR scores of λ^*-H$_G$-QAC and the corresponding changes against λ^*-H-QAC tested on AOL and SnV are recorded in Table 4.11. We find that λ^*-H$_G$-QAC performs better on SnV than on AOL, with higher MRR scores in all cases. However, λ^*-H-QAC consistently outperforms λ^*-H$_G$-QAC as the MRR changes produced by comparing λ^*-H$_G$-QAC against λ^*-H-QAC are always negative.

Another interesting finding is that λ^*-H$_G$-QAC performs very competitive with λ^*-H-QAC, especially on SnV, and the differences are limited (MRR changes: $\sim$1%). This appears to be due to the fact that: (1) λ^*-H$_G$-QAC scores the query similarity on a close character level but confronts the sparseness problem: (2) the number of grams n is artificially fixed, resulting in failure to rank query completions properly.

4.3.8 Performance on Long-tail Prefixes

To answer **RQ6**, we examine the performance of λ^*-H′-QAC on the subsets of the AOL and SnV datasets that only contain long-tail prefixes to compare against the results produced by λ^*-H-QAC under setting $\gamma = 0.5$ in (4.12). We report the results in Table 4.12 in terms of MRR at various number of returned query completions (No.) ranging from 1



(a) AOL **(b) SnV**

Figure 4.8: Performance of λ^*-H-QAC when varying the combination weight γ with a query prefix p length of 1–5 characters for the AOL and SnV query logs.

Table 4.11: MRR scores of λ^*-H$_G$-QAC, as well as MRR changes produced by comparing λ^*-H$_G$-QAC against λ^*-H-QAC (MRR scores presented in Table 4.6), with a query prefix p length of 1–5 characters tested on the AOL and SnV query logs. Significant differences (λ^*-H$_G$-QAC vs. λ^*-H-QAC) are indicated.

	AOL		SnV	
$\#p$	MRR	change	MRR	change
1	0.1213	-0.90%	0.2650	-0.45%
2	0.2066$^{\triangledown}$	-1.21%$^{\triangledown}$	0.3891	-0.41%
3	0.3330$^{\triangledown}$	-1.68%$^{\triangledown}$	0.5309	-0.86%
4	0.4476$^{\blacktriangledown}$	-1.89%$^{\blacktriangledown}$	0.6617$^{\triangledown}$	-1.10%$^{\triangledown}$
5	0.5179	-1.09%	0.7398$^{\triangledown}$	-1.24%$^{\triangledown}$

to 9, including the case of No. = 1, where λ^*-H′-QAC and λ^*-H-QAC display the same performance.

From Table 4.12 we can see that, generally, λ^*-H′-QAC outperforms λ^*-H-QAC in terms of MRR on both datasets. It achieves 1.94% and 2.45% MRR improvements on average over λ^*-H-QAC for all long-tail prefixes in the AOL and SnV subsets, respectively. Moreover, we checked the weights returned by a regression model, which control the contributions from the time-sensitive part and the personalized aspect in λ^*-H′-QAC, i.e., γ and $1-\gamma$ in (4.12), and found that γ is less than 0.5 on both datasets ($\sim$0.42 for AOL and $\sim$0.31 for SnV), implying that personalization is more important for long-tail prefixes. For long-tail prefixes, the final submitted queries often occurred in the current session. In other words, for these cases, repeated query submissions in the same session are often observed.

One particularly interesting observation from Table 4.12 is that λ^*-H′-QAC achieves relatively larger MRR gains over λ^*-H-QAC when the median number of query completions (e.g., No. $=4$ or 5) are returned. This can be attributed to the following: (i) for cases with fewer query completions returned, e.g., No. = 2 or 3, λ^*-H′-QAC achieves similar results, resulting in many draws; (ii) the cases with more query completions returned,

Table 4.12: Performance of λ^*-H-QAC and λ^*-H′-QAC in terms of MRR at various numbers of returned query completions (No.) ranging from 2 to 9 on the subset of AOL and SnV datasets only containing long-tail prefixes. Significant differences (λ^*-H′-QAC vs. λ^*-H-QAC) are indicated.

	AOL		SnV	
No.	λ^*-H-QAC	λ^*-H′-QAC	λ^*-H-QAC	λ^*-H′-QAC
1	1.0000	1.0000	1.0000	1.0000
2	0.7756	0.7842	0.8217	0.8305
3	0.6119	0.6207	0.6412	0.6547
4	0.4701	0.4863$^{\blacktriangle}$	0.5231	0.5398$^{\blacktriangle}$
5	0.4197	0.4318$^{\triangle}$	0.4729	0.4921$^{\blacktriangle}$
6	0.3558	0.3617	0.3927	0.4035$^{\triangle}$
7	0.2987	0.3014	0.3138	0.3198
8	0.2461	0.2492	0.2793	0.2816
9	0.2074	0.2098	0.2239	0.2267

e.g., No. $= 8$ or 9, account for the minority of long-tail prefixes, as shown in Figure 4.5, resulting in limited improvements.

4.3.9 Performance of Modified Hybrid QAC

Finally, we examine the overall performance of λ^*-H′-QAC on the complete datasets (AOL and SnV) to compare against other models. The results in terms of MRR scores are listed in Table 4.6, row 7.

We can see from Table 4.6 that (1) with long-tail prefix detection, our extended hybrid QAC model, i.e., λ^*-H′-QAC, receives the highest MRR scores among the five methods at all lengths of prefix, suggesting that long-tail prefix detection helps boost QAC performance; (2) for some cases, e.g., $\#p = 3$ on AOL, significant improvements at level $\alpha = .01$ are observed by comparing λ^*-H′-QAC against λ^*-TS-QAC, however, which are not seen on the comparisons between λ^*-H-QAC and λ^*-TS-QAC; (3) λ^*-H′-QAC achieves the equal performance with λ^*-H-QAC at $\#p = 1$ because there is no long-tail prefix consisting of only one character.

In general, λ^*-H′-QAC achieves limited improvements over λ^*-H-QAC. This is because the majority of prefixes in the datasets are returned by more than ten candidates, in other words, they are not long-tail prefixes, and in such cases, λ^*-H′-QAC will degenerate to λ^*-H-QAC and then reports the same MRR scores with λ^*-H-QAC.

4.4 Conclusion

Most previous work on query auto completion (QAC) focuses on either time-sensitive maximum likelihood estimation or context-aware similarity. In this chapter we have adopted a combination of the two aspects of the QAC problem, where time-series analysis is used to predict a query's future frequency. To understand a user's personal search

intent, we have extended our time-sensitive QAC method with personalized QAC, which infers the similarity between current requests and preceding queries in a current search session and previous search tasks at the character level. In addition, we have adjusted the model specifically for long-tail prefixes. In particular, we assign an optimal weight $\bar{\gamma}$ in (4.12) to long-tail prefixes after checking their prefix popularity rather than using a fixed weight γ consistent with that assigned to normal prefixes.

As to future work, we intend to have a closer look at the top N candidates returned by the popularity-based ranking method for $N > 10$: how much can we gain from the good candidates that were ranked at lower ranks? Moreover, we aim to transfer our approach to other datasets with long-term query logs, which should help us benefit from queries with longer periodicity than we have access to in the AOL and SnV logs used in our current work. To which degree it is beneficial to diversify QAC results (Cai et al., 2016b)? In addition, we could study a cold-start problem where a user's long-term search logs are unavailable, which could be addressed by using the logs from a group of similar users seen in the training period. A further possible step is to model personalized temporal patterns for active users, especially professional searchers, requiring a generalization from actual query terms to topics or intents. This might help generate a better QAC ranking. In the next chapter, we will turn to explore information from similar query submissions as well as from semantic term closeness, both of which are ignored by current QAC approaches.

5

Learning from Homologous Queries and Semantically Related Terms for Query Auto Completion

We have introduced a time-sensitive personalized query auto completion approach in Chapter 4. In doing so we follow a strict query matching policy when counting the queries, a policy that is shared by most of today's query auto completion models that rank query completions by popularity. However, such models ignore the contributions from so-called *homologous* queries, queries with the same terms but ordered differently or queries that expand the original query. Importantly, homologous queries often express a remarkably similar search intent. Moreover, today's QAC approaches often ignore semantically related terms. We argue that users are prone to combine semantically related terms when generating queries. In this chapter, we propose a learning to rank-based QAC approach, where, for the first time, features derived from homologous queries and semantically related terms are introduced. In particular, we consider: (1) the observed and predicted popularity of homologous queries for a query candidate; and (2) the semantic relatedness of pairs of terms inside a query and pairs of queries inside a session.

First of all, we formally define the two types of homologous queries for a given query $q = (term_1, term_2, \ldots, term_m)$: (1) Given q, a *super query* of q is a query $s_q = (term_1, term_2, \ldots, term_m, term_{m+1}, \ldots, term_L)$ that extends q; (2) A *pseudo-identical query* for q is a query p_q that is a permutation of q. To a certain extent, homologous queries express similar search intents. For instance, in late 2014, for the two queries "*Chile SIGIR*" and "*SIGIR Chile*" (a pseudo-identical query of "*Chile SIGIR*"), the same SERPs should probably be returned. And in 2015 the SERPs for "*Chile SIGIR*" and "*Chile SIGIR 2015*" (a super query of "*Chile SIGIR*") should probably overlap to a very large degree. Based on these examples, we hypothesize that it is advantageous to consider homologous queries as a context resource for QAC.

QAC features inferred from homologous queries are one important innovation that we study in this chapter. A second way of using lexical variations for QAC that we propose is based on semantically related terms. As discussed in the literature, a user's search history usually reveals their search intent, often expressed by the queries or by clicked documents. For instance, Shokouhi (2013) studies the similarity between a query completion and previous queries in both the short-term and long-term history for reranking

Table 5.1: An AOL session example.

	SessionID	UserID	Query	Time
1	821174	1662425	google	20060408, 17:02:46
2	821174	1662425	evanescence	20060408, 17:04:21
3	821174	1662425	ulitimate guitar	20060408, 17:05:13
4	821174	1662425	evanescence videos	20060408, 17:09:44
5	821174	1662425	evanescence videos	20060408, 17:16:23
6	821174	1662425	music videos	20060408, 17:17:31

query completions. And Jiang et al. (2014b) infer features from users' reformulation behavior for reranking query completions. We exploit a similar intuition but operationalize it differently, by considering the semantic relatedness of terms in a query completion and of terms from a query completion and queries previously submitted in the same session. Let us give an example. Consider Table 5.1, which contains a session from the well-known AOL query log. For the sake of the example, let us assume that we have not yet seen the last query (query 6, "music videos") and that it is in fact the initial segment "mus" of this query for which we want to recommend completions. A regular baseline based on query frequency is likely to rank the completion "music" first, as shown in Table 5.2a. If we consider the observed frequency of homologous queries for a candidate, we would return the list seen in Table 5.2b, which is a reranked version of the list in Table 5.2a. Clearly, the queries "music" and "music video" benefit more homologous queries than others as they are now ranked at the top. But if we look in the user's search session (e.g., at query 4 and 5 in Table 5.1), we would see that "videos" is semantically closely related to earlier queries. Based on this insight, the query "music videos" in Table 5.2a is a more sensible completion. By considering the semantic similarity of terms, both inside a candidate and of queries inside a session, we may be able to generate another reranked QAC list, as shown in Table 5.2c. We can see that semantically close queries, e.g., "music videos" and "music video codes," have now been moved to the top of the list.

Table 5.2: Ranked lists of query completions for the prefix "mus".

	(a) Ranked by frequency of candidates.		**(b) Reranked by frequency of homologous queries.**		**(c) Reranked by semantic similarity.**
1	music	1	music	1	music videos
2	music dowloads	2	music videos	2	music video codes
3	music videos	3	music download	3	music dowloads
4	music lyrics	4	music new	4	music new
5	music video codes	5	music codes	5	musicians friend
6	music codes	6	music music dowloads	6	music codes
7	music new	7	music lyrics	7	music dowload
8	music download	8	mustang	8	music lyrics
9	music friend	9	music video codes	9	music
10	mustang	10	musicians friend	10	mustang

Motivated by the examples above, and based on a learning-based QAC model L2R-U that extract features from user behavior (Jiang et al., 2014b), we propose several learning to rank-based QAC approaches, where, for the first time, features derived from predicted popularity, homologous queries and semantically related terms are introduced, respectively. In particular, we consider: (1) the observed and predicted popularity of query completions, which results in the L2R-UP model; (2) the observed and predicted popularity of homologous queries for a query candidate, which results in the L2R-UPH model; (3) the semantic relatedness of pairs of terms inside a query and pairs of queries inside a session, which results in the L2R-UPS model; and (4) all these newly proposed features, which results in the L2R-ALL model. Building on LambdaMART (Burges et al., 2011), these learning to rank (L2R) based QAC models aim to rerank the top N query completions returned by popularity. In doing so, we answer the following research questions:

RQ7 Do the features that describe the observed and predicted popularity of a query completion help boost QAC performance without negatively impacting the effectiveness of user behavior related features proposed in (Jiang et al., 2014b)? That is, how does L2R-UP compare against L2R-U?

RQ8 Do semantic features help improve QAC performance? That is, how does L2R-UPS compare against L2R-UP?

RQ9 Do homologous queries help improve QAC performance? That is, how does L2R-UPH compare against L2R-UP?

RQ10 How does L2R-UPS compare against L2R-UPH? What is the performance gain, if any, if all features are added for learning (L2R-ALL)?

RQ11 What are the principal features developed here for a learning to rank based QAC task?

We address these questions by adding newly extracted features for learning to generate a new QAC ranking model upon a baseline which similarly works in a learning-to-rank framework but less features are considered. We apply these models to experiments on two publicly available query log datasets, finding that features of semantic relatedness and homologous queries are both important and they do indeed help boost QAC performance.

Our contributions in this chapter can be summarized as:

(1) We propose a learning to rank based query auto completion model (L2R-QAC) that exploits contributions from so-called homologous queries for a query completion, in which two kinds of homologous queries are taken into account.

(2) We propose semantic features for QAC, using the semantic relatedness of terms inside a query candidate and of pairs of terms from a candidate and from queries previously submitted in the same session.

(3) We analyze the effectiveness of our L2R-QAC model with newly added features, and find that it significantly outperforms state-of-the-art QAC models, either based on learning to rank or on popularity.

Our experimental results also reveal that semantic features are probably more important than those of homologous queries for QAC tasks in our setting even though the differences are not statistically significant. In other words, query terms are not randomly combined when a searcher formulates a query. Semantically close terms or queries are likely to appear in a query or in a session, respectively.

Table 5.3: Summary of popularity features of query completions (10 features) and the corresponding formulas. The periods considered for the popularity features are $whole$, 1-day, 2-day, 4-day, 7-day for observations and predictions.

Period	Description	Formula
$whole$, 1-, 2-, 4-, 7-day	Observation	*fre(q, period)* in §5.1.1
1-, 2-, 4-, 7-day	Predicted by trend	$\hat{y}_{t_0}(q,i)_{trend}$ as in (5.1)
$whole$	Predicted by periodicity	$\hat{y}_{t_0}(q)_{peri}$ as in (5.2)

The remainder of this chapter is organized as follows. Features for the specific QAC learning problem are described in §5.1. Then, §5.2 presents our experimental setup. In §5.3 we report our experimental results. We conclude in §5.4, where we suggest future research directions.

5.1 Approach

Here, we formally describe the problem of learning to rank (L2R) for query auto completion, and then propose a number of L2R-based QAC models. Four categories of query features that are likely to affect QAC rankings are taken into account: query popularity, user reformulation behavior, features inferred from homologous queries, and semantic features. See Tables 5.3, 5.4 and 5.5 below for a tabular overview.

We use LambdaMART (Burges et al., 2011) to re-rank the top N query completions initially returned by an MPC baseline. Any reasonable L2R algorithm can be employed to obtain our ranking model and will likely yield similar results; we choose LambdaMART because it has been shown to be one of the best algorithms for L2R tasks (Shokouhi, 2013).

5.1.1 Popularity-based Features

Popularity-based features are extracted from two sources: the query candidate (Table 5.3) and its homologous queries (Table 5.4). Both the observed and the predicted query frequency are introduced.

Popularity from observations

We observe the query popularity within various periods, producing pairs of time-sensitive features, denoted as $fre(q, peri)$. The period, identified by a changing time window, can be chosen from the set $\{1\text{-}day, 2\text{-}day, 4\text{-}day, 7\text{-}day, whole\}$, where $whole$ indicates that the popularity is determined based on the entire training query log while the others are collected from logs for the recent 1, 2, 4, or 7 day(s) only. We only focus on these five period options as: (1) our available datasets for learning are not big enough to sensibly consider longer intervals, e.g., the MSN log that we use only covers a one month period; (2) the sliding windows chosen as part of the time-sensitive approach to QAC in (Whiting and Jose, 2014) can successfully reveal recent trends of query popularity.

Table 5.4: Summary of popularity features of homologous queries for query completions (60 features) and the corresponding formulas. There are two categories of homologous queries, *super queries* and *pseudo-identical queries*. The weight of super queries can be determined using either query term overlap (§5.1.2) or common prefix weighting (§5.1.2). We consider two choices for each feature: maximum or summation as described in §5.1.2.

Period	Impact	Description	Formula
Super query			
$whole$, 1, 2, 4, 7-day	w in (5.5) or (5.6)	Observation	$fre(q', peri)$ in §5.1.1
1-, 2-, 4-, 7-day	w in (5.5) or (5.6)	Predicted by trend	$\hat{y}_{t_0}(q', i)_{tre}$ in (5.1)
$whole$	w in (5.5) or (5.6)	Predicted by periodicity	$\hat{y}_{t_0}(q')_{per}$ in (5.2)
Pseudo-identical query			
$whole$, 1, 2, 4, 7day	$w = 1$	Observation	$fre(q', peri)$ in §5.1.1
1-, 2-, 4-, 7-day	$w = 1$	Predicted by trend	$\hat{y}_{t_0}(q', i)_{tre}$ in (5.1)
$whole$	$w = 1$	Predicted by periodicity	$\hat{y}_{t_0}(q')_{per}$ in (5.2)

Popularity from predictions

As introduced in Chapter 4, we generate features based on predicted query popularity according to the recent trend and according to cyclic phenomena. Specifically, we first detect the trend of query q's popularity from the first-order derivative of its daily count $C(q,t)$ observed at different time points t, and then predict its future popularity at day t_0, denoted by $\hat{y}_{t_0}(q,i)_{tre}$, based on the observation of each preceding i-th day as:

$$\hat{y}_{t_0}(q,i)_{tre} = y_{t_0-i}(q,i) + \int_{t_0-i}^{t_0} \frac{\partial C(q,t)}{\partial t} \mathrm{d}t, \tag{5.1}$$

where $y_{t_0-i}(q,i)$ is the observed frequency of q at the preceding i-th day. For simplicity, similar to the choices of $period$ in §5.1.1, we consider four options for i, i.e., $i \in \{1,2,4,7\}$.

In addition, we predict the query popularity according to its periodicity of query volume, denoted by $\bar{y}_{t_0}(q)_{per}$. We detect cyclic phenomena of query popularity at a per hour level and then produce an aggregated query popularity at a per day level. We smooth $\bar{y}_{t_0}(q)_{per}$ by simply averaging the recent M observations y_{t_p} at the preceding time points $t_p = t_0 - 1 \cdot T_q, \ldots, t_0 - M \cdot T_q$ in the log:

$$\hat{y}_{t_0}(q)_{per} = \frac{1}{M} \sum_{m=1}^{M} y_{t_0 - m \times T_q}(q), \tag{5.2}$$

where T_q denotes query q's periodicity. For detecting cyclic aspects of q's frequency, we use autocorrelation coefficients (Chatfield, 2004), which measure the correlation between N_s successive count observations $C(q,t)$ at different times $t = 1, 2, \ldots, N_s$ in the query log. The correlation is computed between a time series and the same series lagged by i

time units:

$$r_i = \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)(C(q,t+i)-\bar{x}_2)}{(\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x}_1)^2)^{\frac{1}{2}}(\sum_{t=i+1}^{N_s}(C(q,t+i)-\bar{x}_2)^2)^{\frac{1}{2}}}, \tag{5.3}$$

where $\bar{x}_1$ is the mean of the first $N_s - i$ observations and $\bar{x}_2$ is the mean of the last $N_s - i$ observations. For reasonably large N_s, the denominator in (5.3) can be simplified by approximation. First, the difference between the sub-period means $\bar{x}_1$ and $\bar{x}_2$ can be ignored. Second, the difference between summations over observations 1 to $N_s - i$ and $i + 1$ to N_s can be ignored. Consequently, r_i can be approximated as follows:

$$r_i \approx \frac{\sum_{t=1}^{N_s-i}(C(q,t)-\bar{x})(C(q,t+i)-\bar{x})}{\sum_{t=1}^{N_s}(C(q,t)-\bar{x})^2}, \tag{5.4}$$

where $\bar{x} = \sum_{t=1}^{N_s} C(q,t)$ is the overall mean. A more elaborate discussion can be found in §4.1.1.

5.1.2 Weighting Homologous Queries

Now, given a query q, $Hom(q)$, the set of homologous queries for q, consists of all super queries s_q and pseudo-identical queries p_q of q found in the query logs, which have been issued before q was submitted. We extract the same popularity features discussed in §5.1.1 for each homologous query $q'_c \in Hom(q_c)$ for a candidate q_c. We are not going to use all homologous queries q'_c for a candidate query q_c with the same weight when generating the final popularity features of homologous queries. Instead, we measure how similar q'_c and q_c are with weight w to be able to weigh the contribution of q'_c—this will allow us to capture the popularity of q_c from a homologous query q'_c.

After discounting the popularity of homologous queries, we calculate the maximal and aggregated values from all homologous queries as features. Next, we will introduce two approaches for discounting.

Query term overlap weighting

Clearly, the more terms the homologous query q'_c and q_c share, the more relevant they could be to each other. Let $q'_c \cap q_c$ refer to the set of common terms in q'_c and q_c. We multiply the term overlap ratio as a discount $Discount(q'_c, q_c)$ with the popularity of super query s_q of candidate q_c:

$$w \leftarrow Discount(s_q, q_c) = \frac{|s_q \cap q_c|}{|s_q|}, \tag{5.5}$$

where $|\cdot|$ returns the number of terms of the input term set.

For pseudo-identical queries p_q for q_c, we set $Discount(p_q, q_c) = 1$ as we assume that they enjoy the same search popularity.

Table 5.5: Summary of semantic features of candidate (14 features) and the corresponding formulas. We consider two choices for each feature: maximum or summation and there are 2 sources for determining the word2vec scores, the GoogleNews corpus or one of our query logs (AOL or MSN).

Description	Formula
Word2vec score from GoogleNews or query logs	word2vec in §5.1.3
Lexical similarity score from query logs	$f_{WordSimMax}$ and $f_{WordSimSum}$ in §5.1.3
Term pair likelihood ratio from query logs	f_{LLRMax} and f_{LLRSum} as (5.7)
Term pair cooccurrence frequency from query logs	f_{cof_max} and f_{cof_sum} in §5.1.3
Lexical query similarity score from query logs	$f_{QueSimMax}$ and $f_{QueSimSum}$ in 5.1.3
Temporal relation from Query logs	$f_{TemRelMax}$ and $f_{TemRelSum}$ as in (5.8)

Common prefix weighting

In addition to the query term set overlap discounting, we may also posit that a longer common prefix matters more than shorter ones. This can be captured by introducing an impact factor based on the length of the overlapping prefix. Thus, we discount the popularity of super query s_q with $Impact(s_q, q_c)$ as:

$$w \leftarrow Impact(s_q, q_c) = \frac{\|CommonPrefix(s_q, q_c)\|}{\|s_q\|}, \tag{5.6}$$

where $\|\cdot\|$ returns the number of characters of the input string. Rather than emphasizing the overlapping terms, this emphasizes the longest common prefix. For pseudo-identical queries p_q of q_c we assign an impact $Impact(p_q, q_c) = 1$.

5.1.3 Semantic Features

We use semantic information in two ways. Our example in the introduction to this chapter suggests that queries semantically related to queries submitted earlier in a session may be more likely to be good completion candidates. We also estimate the semantic relatedness of words occurring in a query: we assume that semantically more coherent queries may be better query candidates. We start with the latter.

Semantic relatedness in queries

We use several ways of computing semantic relatedness between terms in a query. To begin, we use a lexical similarity method that combines POS tagging, Latent semantic analysis and WordNet to determine term-level similarity (Han et al., 2013). Given two words, it returns a string representing a number between 0.0 and 1.0 with 1.0 indicating absolutely similar. We use it to capture the word similarity, resulting in two features $f_{WordSimMax}$ and $f_{WordSimSum}$ obtained by maximizing and summing all similarity scores of possible term pairs in a query, respectively.[1]

[1]We apply a web API released by UMBC that can be used to return the semantic similarity score between words, `http://swoogle.umbc.edu/SimService/api.html`.

Next, we consider the query term pair likelihood ratio. Jones et al. (2006) observe that frequent query pairs from search sessions can be found by statistical hypothesis testing. Given two queries, the likelihood ratio (LLR) between them can be calculated. LLR testing is performed to see whether their co-occurrence in a search session is statistically significant. Similarly, we argue that frequently co-occurring term pairs in a query could be semantically related and introduce the LLR score to capture the semantic relatedness of term pairs inside a query. More specifically, assuming that there are two terms $term_1$ and $term_2$ in a query, we calculate the LLR score for this term-pair as:

$$LLR(term_1, term_2) = -2 \log \frac{L(term_2 \mid \neg term_1)}{L(term_2 \mid term_1)}, \tag{5.7}$$

where $L(term_2 \mid \neg term_1)$ denotes the number of queries containing $term_2$ but without $term_1$, and $L(term_2 \mid term_1)$ indicates the volume of queries containing both $term_1$ and $term_2$. High LLR scores are assumed to indicate semantic relatedness. We calculate LLR scores for all possible term pairs in a query q (excluding stop words) to measure the semantic relatedness. Then, we return the maximal and aggregated LLR scores of all term pairs as semantic features, resulting in f_{LLRMax} and f_{LLRSum}. In addition, we use the larger LLR score of a term pair $(term_i, term_j)$ in a query q in spite of the term order, and then produce the maximal and aggregated LLR scores as partial semantic features, indicated as

$$f_{cof\_max} = \max_{term_{i,j} \in q} (\max(L(term_i \mid term_j), L(term_j \mid term_i)))$$

and

$$f_{cof\_sum} = \sum_{term_{i,j} \in q} \max(L(term_i \mid term_j), L(term_j \mid term_i)).$$

Our third way of operationalizing semantic relatedness between query terms builds on the method described in (Mikolov et al., 2013a) and known as word2vec, where vector representations of words are learned from large amounts of unstructured text resources, e.g., GoogleNews. Representations of words as continuous vectors have been shown to capture meaningful semantic word regularities (Mikolov et al., 2013b). We use the idea to capture semantic relatedness between query terms. The training objective of the Skip-gram model is to learn word vector representations that are good at predicting the surrounding words by maximizing the average log probability

$$\frac{1}{T_r} \sum_{t=1}^{T_r} \sum_{-c_s <= j <= c_s, j \neq 0} \log P(term_{t+j} \mid term_t),$$

with inputting a sentence of terms $term_1$, $term_2$, ..., $term_{T_r}$, where c_s is the size of training context. This setup allows us to assign a word2vec score, learnt from GoogleNews, to each term pair in a query. In this manner, we produce $f_{W2vGooMax}$ and $f_{W2vGooSum}$ by maximizing and summing scores of all term pairs.

As the trained word representation is a data-driven approach, which is highly dependent on the text resource and limited by its inability to represent idiomatic queries or rare words unseen in the text resources, we also train a local Skip-gram model on a

query log dataset to represent query terms as a compensation and then calculate a local log-based semantic relatedness score. In other words, we input queries in the query logs one by one as a sequence of training terms instead of sentences in the unstructured text resource. Based on the word2vec model learnt from the query logs, the maximal and aggregated word2vec scores of term pairs are returned, resulting in features $f_{W2vLogMax}$ and $f_{W2vLogSum}$.

Semantic relatedness in sessions

So far, we have considered the semantic relatedness between words inside a query. Next, we consider the semantic relatedness between query candidates and previous queries in the same session. We use alternative methods to capture this relation.

First, we capture query relatedness between query candidates and previous queries in a session at the lexical level, by using the combination of latent semantic analysis, POS tagging and WordNet mentioned before; this results in the features $f_{QueSimMax}$ and $f_{QueSimSum}$.

Second, following (Chien and Immorlica, 2005), we investigate query relatedness using temporal correlations. In other words, two queries are assumed to be semantically related in this sense if their popularities behave similarly over time. We employ Pearson's correlation coefficient, commonly represented by the letter r, to capture this notion of similarity between candidate q_c and previous query q_x. Similar to (5.4), we can obtain a formula for $r(q_c, q_x)$ as:

$$r(q_c, q_x) = \frac{1}{n_u} \sum_{i=1}^{n_u} \left(\frac{q_{c,i} - \mu(q_c)}{\delta(q_c)} \right) \left(\frac{q_{x,i} - \mu(q_x)}{\delta(q_x)} \right), \tag{5.8}$$

where the frequency of query q_c (or q_x) over n_u days is an n_u-dimensional vector $q_c = (q_{c,1}, q_{c,2}, \ldots, q_{c,n_u})$ with $\mu(q_c)$ and $\delta(q_c)$ indicating the mean frequency and the standard deviation of its frequency, respectively. The correlation $r(q_c, q_x)$ of two queries q_c and q_x is a standard measure of how strongly two queries are linearly related. It always lies between $+1$ and -1, with $+1$ implying an exactly positive linear relationship between them and -1 for an absolutely negative linear relationship. Again, we calculate the maximal and aggregated query relatedness scores of all query pairs as temporal semantic features, resulting in $f_{TemRelMax}$ and $f_{TemRelSum}$, respectively.

5.1.4 Feature Summary

We also make use of user reformulation behavior features following (Jiang et al., 2014b), derived at three levels: term-, query- and session-level. Please refer to (Jiang et al., 2014b) for details. The use of features of homologous queries and semantic relatedness for QAC is a new contribution of this chapter. In total, we use 127 features including those from (Jiang et al., 2014b).

Our features are slotted into three categories: popularity, user reformulation behavior, and semantic relatedness. For query popularity, we collect evidence from the candidate and its homologous queries, based on observations and predictions. For user behavior features, we directly use those from (Jiang et al., 2014b). Regarding homologous

queries, there are 2 categories, 5 period options for observation (4 for prediction by trend), 2 weighting schemes (only for super queries) and 2 calculations, accounting for a total of 60 features, i.e., 30 features by observation and 30 features by prediction (24 from trend and 6 from periodicity). Referring to the candidate's popularity, recent observations and predictions are collected by simply changing the time period, yielding $5 + 4 + 1 = 10$ features. For the semantic relatedness features, 2 sources are used to calculate the word2vec score: GoogleNews and a local query log, i.e., AOL (MSN) queries are scored by the corresponding word2vec model trained on the AOL (MSN) logs; 2 calculations are adopted for generating popularity features of homologous queries and semantic features: maximization and summation; 2 term-pair significant scores are included; semantic relatedness is measured on word- and query-level, resulting in 14 features.

Accordingly, in total, we develop 70 ($= 60 + 10$) features for popularity, 14 features for semantic relatedness and 43 features for user reformulation behavior, yielding a total of $70 + 14 + 43 = 127$ features for our L2R-based QAC approach.

5.2 Experiments

In this section we detail our experimental setup. §5.2.1 summarizes the proposed models trained on different features; §5.2.2 describes our datasets; we detail further experimental settings and parameters in §5.2.3.

5.2.1 Model Summary

To examine the contribution from each specific feature source, we specify six L2R-QAC variations depending on the features used: L2R-U, -UP, -UPS, -UPH, -ALL and -TOP; see Table 5.6.

For comparison, we consider several QAC baselines: (1) the Most Popular Completion (MPC) model, which is based on query frequencies in the whole log and has been discussed in §4.2.1, referred to as MPC-ALL (Bar-Yossef and Kraus, 2011); (2) an MPC based QAC method, using frequency within a recent time window, denoted as MPC-R (Whiting and Jose, 2014), which is state-of-the-art. Here we set $R = 7$ days as the time window because performance peaks with this setting (Whiting and Jose, 2014); (3) a recent L2R-based QAC model considering user reformulation behavior features (Jiang et al., 2014b), denoted as L2R-U. The former two baselines are popularity-based while L2R-U is a L2R-based QAC model.

To select a single baseline against which we compare our newly introduced models, we compare the performance of the three baselines just listed and report the results in Table 5.7. For both datasets, L2R-U outperforms the other two baselines. For instance on AOL, it results in more than 8% and 5% improvements in MRR over MPC-ALL and MPC-R, respectively, after typing one character as prefix. Similar results can be found on the MSN dataset. Hence, we select L2R-U as the single baseline for comparison with our proposed models in later experiments.

Table 5.6: An overview of the QAC models discussed in the chapter.

Model	Description	No. of features	Source
Baselines			
MPC-ALL	Ranking query completions according to their past popularity in the whole log	–	(Bar-Yossef and Kraus, 2011)
MPC-R	Ranking query completions according to their past popularity within recent *R* days	–	(Whiting and Jose, 2014)
L2R-U	Learning to rank query completions using user reformulation behavior features	43	(Jiang et al., 2014b)
Our models			
L2R-UP	Extending L2R-U by adding 10 popularity features of the same query	$43 + 10 = 53$	This chapter
L2R-UPS	Extending L2R-UP by adding 14 semantic features of the same query	$53 + 14 = 67$	This chapter
L2R-UPH	Extending L2R-UP by adding 60 popularity features of homologous queries	$53{+}60 = 113$	This chapter
L2R-ALL	All features are considered for learning to rank query completions	$67{+}60 = 127$	This chapter
L2R-TOP	Extending L2R-UP by only adding top ten important features newly developed here	$53 + 10 = 63$	This chapter

5.2.2 Datasets

In this chapter, we implement our models on the MSN query log (Craswell et al., 2009) as well as the AOL query log (Pass et al., 2006), both of which have been discussed in Chapter 3. Both datasets are publicly available. For consistency, we partition each log into two parts: a training set consisting of 75% of the query log and a test set consisting of the remaining 25% in terms of time period. We also use the temporally last 10% samples in the training set as validation set for LambdaMART. Traditional *k*-fold cross-validation is not applicable to a streaming sequence since it would negate the temporal order inherent in the data (Gama et al., 2014). Queries in the training set were submitted before May 8, 2006 in the AOL dataset and before May 24, 2006 in the MSN dataset.

We divide the queries into sessions by 30 minutes' inactivity[2] and only English queries appearing in both two partitions were kept. Importantly, in our experimental

[2]Only applied on the AOL dataset as the MSN dataset provides session IDs.

Table 5.7: Selecting our baseline. The performance of various baselines in terms of MRR and SR@K, tested on the AOL and MSN datasets after typing one character as prefix. The best performing baseline in each row is highlighted.

Dataset	Metric	MPC-ALL	MPC-R	L2R-U
AOL	MRR	0.6157	0.6348	**0.6682**
	SR@1	0.4532	0.4643	**0.4815**
	SR@2	0.5914	0.6038	**0.6256**
	SR@3	0.7016	0.7121	**0.7304**
MSN	MRR	0.6305	0.6498	**0.6821**
	SR@1	0.4702	0.4757	**0.4876**
	SR@2	0.6083	0.6276	**0.6385**
	SR@3	0.7251	0.7368	**0.7437**

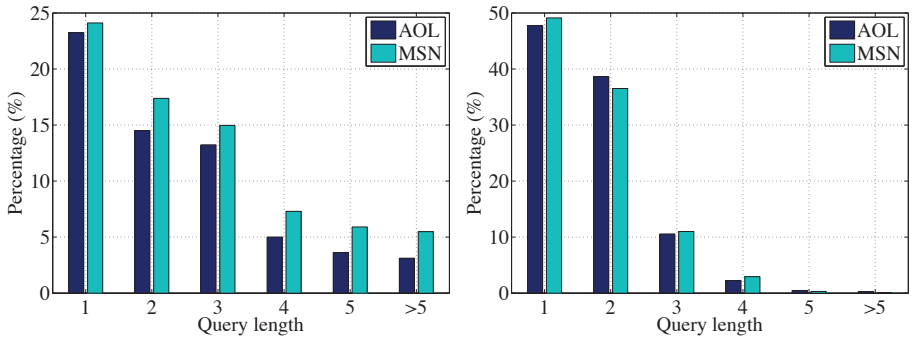
Table 5.8: Statistics of the AOL and MSN datasets used. The "$\star$" in # Prefix-$\star$ indicates the length of the prefix in characters.

	AOL		MSN	
Variables	Training	Test	Training	Test
# Queries	3,808,083	1,571,346	3,784,925	1,402,308
# Unique queries	452,980	452,980	304,943	304,943
# Sessions	1,149,528	465,302	674,717	256,512
Queries / session	3.31	3.38	5.60	5.46
# All prefixes	8,783,957	3,260,130	4,995,213	1,751,158
# Prefix-1	605,710	209,650	427,502	141,925
# Prefix-2	1,175,087	405,857	749,821	249,065
# Prefix-3	1,954,285	707,580	1,134,539	387,633
# Prefix-4	2,433,385	916,976	1,320,529	470,286
# Prefix-5	2,615,490	1,020,067	1,362,822	502,249

design we follow Jiang et al. (2014b) and focus on sessions with at least two queries. By doing so, we can extract user behavior related features. Like previous session context-based QAC approaches (Bar-Yossef and Kraus, 2011; Jiang et al., 2014b), we set the prefix in our experiments to be the first 1–5 character(s) of queries in a session. To obtain our training and test sets, we remove input prefixes for which the ground truth (see below) is not included in the top ten query completions returned by MPC at the time point of querying; this too follows previous QAC work and is a commonly used methodology in QAC tasks (Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013). Table 5.8 details the statistics of our processed datasets.

The ground truth for a QAC task is defined as follows. Given a search session with T queries, i.e., $\{q_1, q_2, \ldots, q_{T-1}, q_T\}$, we want to predict each intended query $q_i, i = \{1, 2, \ldots, T\}$ at position i of a session after typing its prefix p. Then, a query q is a *correct completion* of this prefix p of q_i if, and only if, $q = q_i$.

Next we take a closer look at the processed datasets so to be able to report the ratio of queries at various lengths in words and of queries that have homologous queries. As



(a) The ratio of queries possessing homologous queries.

(b) Distribution of queries with various number of terms.

Figure 5.1: Statistics of queries at various query lengths (in words) for the AOL and MSN datasets, respectively.

shown in Figure 5.1a, generally, for both datasets, nearly one quarter of the one-term queries have homologous queries, solely contributed by their super queries. For two- and three-term queries, super queries and pseudo-identical queries contribute similarly, however for longer queries (> 3 words), the latter dominate the contribution. In addition, as plotted in Figure 5.1b, more than half of all queries contain more than one word, which supports the feasibility for semantic features between words inside a query.

5.2.3 Settings

As introduces in §4.1.1, for time-sensitive query popularity prediction, we count queries per hour to detect the periodicity and aggregate the hour-predictions within the same day to generate the day-volume. For smoothing in (5.2), we set $M = 3$ as it performs best when M changes from 1 to 5 in our trials. Before we run our L2R-based QAC experiments, we are given a list of top N query completions by the traditional MPC approach; we set $N = 10$ as this is commonly used by many web search engines and published QAC work (Cai et al., 2014a,b; Jiang et al., 2014b; Shokouhi, 2013). We use the LambdaMART learning algorithm for ranking query completions across all experiments (Burges et al., 2011).

5.3 Results and Discussion

In §5.3.1, we examine the performance of L2R-UP, which we follow with a section about the contribution of semantic features in §5.3.2. We examine the performance of L2R-UPH in §5.3.3 with features of homologous queries added to L2R-UP. Next, §5.3.4 details the performance of L2R-ALL learnt from all features above and §5.3.5 provides an analysis of feature importance. Finally, §5.3.6 zooms in on the impact of query position on QAC performance.

5.3.1 Effect of Query Popularity

Since information about the past popularity of query completions can be generated offline before ranking while the true popularity is unavailable at runtime, we develop the popularity features according to their previously observed frequency within various periods in the query logs, known as time-sensitive popularity features. In contrast, we produce the predicted query popularity as features based either on recent trends or on cyclic patterns. In this section, we compare the performance of L2R-UP with that of the baseline.

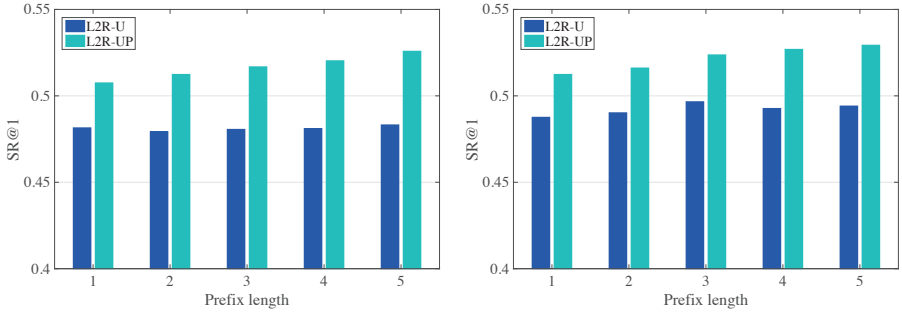
Table 5.9 includes the results on two datasets, i.e., the AOL and MSN datasets, after entering prefixes consisting of 1 to 5 characters. On each dataset, L2R-UP generally performs better than the baseline (L2R-U) in terms of MRR. When we take a closer

Table 5.9: Performance in terms of MRR for the QAC task, at a prefix length $\#p$ ranging from 1 to 5 characters on the AOL and MSN datasets. For each dataset the best performer per row is highlighted. Statistically significant differences are determined against the baseline.

	AOL					
$\#p$	Baseline	L2R-UP	L2R-UPS	L2R-UPH	L2R-ALL	L2R-TOP
1	0.6682	0.6764	0.6871$^{\triangle}$	0.6847$^{\triangle}$	**0.6977**$^{\blacktriangle}$	0.6913$^{\triangle}$
2	0.6631	0.6815$^{\triangle}$	0.6939$^{\blacktriangle}$	0.6898$^{\triangle}$	**0.7024**$^{\blacktriangle}$	0.6980$^{\blacktriangle}$
3	0.6654	0.6853$^{\triangle}$	0.7001$^{\blacktriangle}$	0.6910$^{\triangle}$	**0.7081**$^{\blacktriangle}$	0.7042$^{\blacktriangle}$
4	0.6673	0.6921$^{\triangle}$	0.7094$^{\blacktriangle}$	0.6981$^{\blacktriangle}$	**0.7144**$^{\blacktriangle}$	0.7127$^{\blacktriangle}$
5	0.6704	0.6986$^{\blacktriangle}$	0.7186$^{\blacktriangle}$	0.7059$^{\blacktriangle}$	**0.7215**$^{\blacktriangle}$	0.7201$^{\blacktriangle}$
	MSN					
$\#p$	Baseline	L2R-UP	L2R-UPS	L2R-UPH	L2R-ALL	L2R-TOP
1	0.6821	0.6933	0.7028$^{\triangle}$	0.7011$^{\triangle}$	**0.7136**$^{\blacktriangle}$	0.7084$^{\triangle}$
2	0.6847	0.6971	0.7112$^{\triangle}$	0.7048$^{\triangle}$	**0.7204**$^{\blacktriangle}$	0.7183$^{\blacktriangle}$
3	0.6915	0.7080$^{\triangle}$	0.7225$^{\blacktriangle}$	0.7135$^{\triangle}$	**0.7287**$^{\blacktriangle}$	0.7251$^{\blacktriangle}$
4	0.6873	0.7113$^{\triangle}$	0.7260$^{\blacktriangle}$	0.7164$^{\blacktriangle}$	**0.7314**$^{\blacktriangle}$	0.7300$^{\blacktriangle}$
5	0.6895	0.7212$^{\blacktriangle}$	0.7366$^{\blacktriangle}$	0.7263$^{\blacktriangle}$	0.7416$^{\blacktriangle}$	**0.7487**$^{\blacktriangle}$

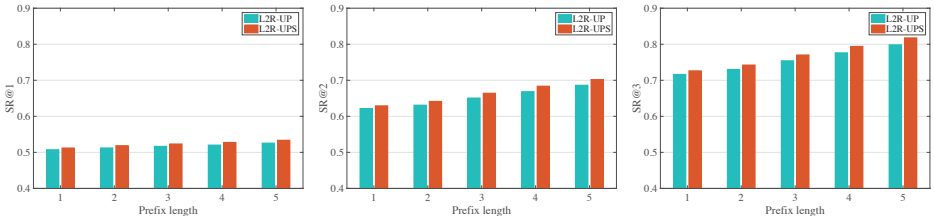
look at the results across all prefixes, reported in MRR scores in Table 5.9, L2R-UP is considerably more effective on longer prefixes as it produces larger MRR improvements over the baseline on long prefixes, e.g., $\# = 4$ or 5. Longer prefixes notably reduce the space of possible query completions, which simplifies the problem. In addition, L2R-UP shows a monotonous increase in MRR as the prefix length goes up. However, the baseline shows a bit fluctuation in terms of MRR as the prefix length changes. Expectedly, L2R-UP always receives the higher MRR scores compared to the baseline.

Next, we compare L2R-UP against the baseline (L2R-U) in terms of SR@1 and plot the results in Figure 5.2. We find that, at each prefix length, for more than half of the test prefixes, L2R-UP returns the final submitted query at the first position in the QAC ranking list because all SR@1 scores achieved by L2R-UP are higher than 0.5; L2R-U receives a lower SR@1 score than 0.5 on both datasets; long prefixes invariably result in higher SR@1 scores than short ones. From these findings, we conclude that the observed



(a) AOL **(b) MSN**

Figure 5.2: QAC performance in terms of SR@1 observed for L2R-U and L2R-UP, tested on the AOL and MSN datasets.



(a) SR@1 **(b) SR@2** **(c) SR@3**

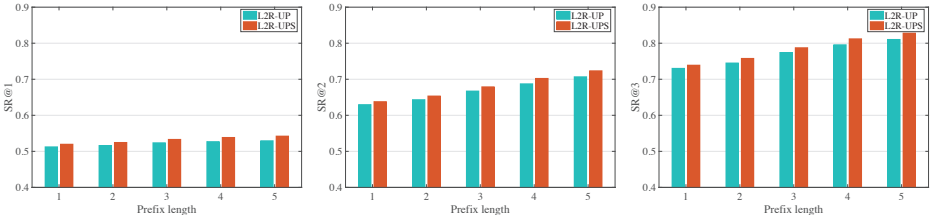
Figure 5.3: QAC performance of L2R-UP and L2R-UPS tested on the AOL dataset at various prefix lengths (in characters), in terms of SR@1, SR@2 and SR@3, respectively.

and predicted popularity features of query candidates indeed help generate better QAC rankings when embedded into a learning to rank framework.

5.3.2 Effect of Semantic Features

To answer **RQ8**, we learn our L2R-UPS model by extending L2R-UP with additional 14 semantic features. The MRR scores of L2R-UPS are listed in Table 5.9; we also plot the scores in terms of other metrics (SR@K, K = 1, 2, 3) of L2R-UP and L2R-UPS in Figure 5.3 and 5.4, tested on the AOL and MSN datasets, respectively, with varying lengths of query prefix from 1 to 5.

Generally, we find that L2R-UPS beats L2R-UP for all cases on both datasets in terms of MRR and SR@K (K = 1, 2, 3). In particular, on the AOL dataset, the MRR improvements of L2R-UPS over L2R-UP are statistically significant (at level $\alpha = .05$) for most cases, e.g., at $\#p$ = 4 and 5; however, on the MSN dataset, most of the improvements are not significant except at $\#p$ = 5 (at level $\alpha = .05$). This is due to the fact that compared to the AOL dataset, the MSN dataset contains far more short queries, resulting in difficulties in capturing the term-pair semantic relatedness. In contrast, compared to the



(a) SR@1 **(b) SR@2** **(c) SR@3**

Figure 5.4: QAC performance of L2R-UP and L2R-UPS tested on the MSN dataset at various prefix lengths (in characters), in terms of SR@1, SR@2 and SR@3, respectively.

baseline (L2R-U), L2R-UPS shows significant MRR improvements for all cases on both datasets. For instance, on the AOL dataset, significant MRR improvements are observed at level $\alpha = .01$ for prefix length $\#p$ = 2 to 5 and at level $\alpha = .05$ for prefix length $\#p$ = 1 by comparing L2R-UPS against the baseline, respectively.

Clearly, the gains in MRR of L2R-UPS over L2R-UP are larger for longer prefixes. E.g., on the AOL dataset, L2R-UPS achieves a 2.86% improvement over L2R-UP at $\#p = 5$ and only a 1.58% improvement at $\#p = 1$ in terms of MRR. Similar observations can be made for other metrics. The results on the MSN dataset show a similar behavior even though the gaps are smaller. With longer prefixes, query candidates are more likely composed of multiple terms, which helps to extract semantic features. Regarding the outcomes in terms of SR@K, as shown in Figure 5.3 and 5.4, for the majority of cases, L2R-UPS can return the correct query in top 3 of the QAC list as the scores in terms of SR@3 are higher than 0.8 on both datasets. Hence, we conclude that the semantic relatedness features can help generate "good" queries, in which terms are semantically close to each other.

5.3.3 Effect of Homologous Queries

Next, we turn to **RQ9** and examine the contribution from features of homologous queries for the candidate. Recall that the resulting model is called L2R-UPH (see Table 5.6). We generate the QAC rankings for each prefix; the MRR scores are included in Table 5.9, column 5. We see that L2R-UPH significantly outperforms the baseline, on both datasets, resulting in near 5% improvements in terms of MRR for long prefixes, e.g., for $\#p = 4$ or 5, but less for short prefixes, e.g., $\#p = 1$. Generally, L2R-UPH achieves 4.4% and 4.1% improvements over the baseline on the AOL and MSN datasets, respectively, in terms of MRR.

Additionally, we compare L2R-UPH against L2R-UP in terms of MRR and SR@1 and report the relative changes in Table 5.10. Across the board, L2R-UPH outperforms L2R-UP in terms of MRR and SR@1. L2R-UPH achieves an average improvement in MRR scores around 1.2% on AOL and 0.9% on MSN over L2R-UP, respectively. For all cases, the improvement of L2R-UPH over L2R-UP is not statistically significant. Interestingly, the gains in MRR of L2R-UPH over L2R-UP are larger for shorter prefixes

Table 5.10: Changes in MRR and SR@1 scores between L2R-UPH and L2R-UP at varying prefix lengths on the AOL and MSN datasets.

	AOL		MSN	
$\#p$	MRR	SR@1	MRR	SR@1
1	+1.23%	+1.30%	+1.13%	+1.21%
2	+1.22%	+1.19%	+1.10%	+1.09%
3	+0.83%	+0.93%	+0.78%	+0.90%
4	+0.87%	+0.86%	+0.72%	+0.82%
5	+1.04%	+1.07%	+0.71%	+0.79%

(e.g., $\#p = 1$ or 2), which differs from the outcomes of comparing L2R-UPS against L2R-UP where obvious MRR gains are found for long prefixes. We believe that this is due to the fact that shorter prefixes result in more ambiguous and shorter candidates, leading to a higher probability for query completions to possess homologous queries from which more information can be gleaned.

Next, we zoom in on the difference between L2R-UPS and L2R-UPH. L2R-UPS tends to outperform L2R-UPH in terms of MRR at all prefix lengths (see Table 5.9, column 4 vs. 5), resulting in an average improvement over L2R-UPH of around 1.5% on the AOL dataset and 1.3% on the MSN dataset. The differences increase as the prefix becomes longer. Hence, even though most differences are not statistically significant, semantic features are probably more important than those of homologous queries for QAC tasks in our setting.

In sum, homologous queries help improve the ranking of query completions, reflecting the fact that searchers occasionally modify queries by changing the term order or adding terms. In addition, compared to the contribution from homologous queries, semantic features provide a bigger contribution to L2R-based QAC tasks as L2R-UPS is more effective than L2R-UPH on both datasets (AOL and MSN).

5.3.4 Performance of L2R-ALL

For research question **RQ10** we examine whether our L2R-ALL model learnt from all discussed features can help boost QAC ranking performance. The MRR scores achieved by L2R-ALL are listed in Table 5.9, column 6, on the AOL and MSN datasets. Clearly, for both datasets, L2R-ALL is considerably more effective than L2R-UPS and L2R-UPH, especially for short prefixes.

In addition, we examine the difference in MRR scores between L2R-ALL, L2R-UPS and L2R-UPH, respectively. For both datasets, the improvement of L2R-ALL over L2R-UPS is not significant. However, for all cases on AOL except $\#p = 1$ and 2, L2R-ALL significantly outperforms L2R-UPH at the $\alpha = .05$ level; for MSN, significant improvements of L2R-ALL over L2R-UPH are observed, except for $\#p = 1$ at level $\alpha = .05$. Generally, L2R-ALL achieves a 1.2% improvement in terms of MRR over L2R-UPS on both datasets. Compared to L2R-UPH, L2R-ALL shows a 2.3% MRR improvement in general. Regarding the comparisons to the baseline, L2R-ALL achieves significant improvements in terms of MRR at the $\alpha = .01$ level for all prefix lengths

Table 5.11: Per prefix bakeoff, in terms of reciprocal rank: L2R-ALL vs. other models. The ratios (%) of test prefixes at all lengths for which L2R-ALL loses against the corresponding model listed in column 1 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which L2R-ALL wins have a green background.

Model	AOL			MSN		
Baseline	4.21	52.17	43.62	5.48	54.30	40.22
L2R-UP	9.13	54.92	35.95	10.10	56.53	33.37
L2R-UPS	18.37	54.40	27.23	12.24	54.79	32.97
L2R-UPH	9.35	53.11	37.54	10.46	55.91	33.63

on both datasets. In particular, L2R-ALL achieves an average 6.8% and 6.3% MRR improvement on AOL and MSN, respectively.

Next, we check the QAC ranking performance per prefix and list the ratio of test prefixes at all lengths for which L2R-ALL loses against, equals or outperforms the corresponding models; see Table 5.11. We can see that, on both datasets, L2R-ALL presents a majority of draws with the other models. Actually, the draws are often observed on prefixes for which all models return the correct query submission at the top positions, e.g., 1 or 2. That is why these models receive high MRR scores (see Table 5.9). Additionally, compared with the other three models in Table 5.11, i.e., Baseline, L2R-UP and L2R-UPH, the L2R-UPS model beats L2R-ALL more often, especially on the AOL dataset, usually on long prefixes, e.g., $\#p = 4$ or 5.

5.3.5 Feature Sensitivity Analysis

Finally, we analyze the relative importance of our newly developed features to answer **RQ11**. Following the methodology used by Agichtein et al. (2008), the top ten most significant features on each dataset used for learning, according to a χ^2 test, are reported in Table 5.12.

We see that the word2vec score returned by the word2vec model on the query logs, with the maximal value of all term pairs in a query, appears to be the most important feature. Generally, semantic relatedness features are more important than features of homologous queries, as they are ranked higher and account for the majority of the top ten features. Popularity features of pseudo-identical (PI) queries are notably more helpful for QAC than super queries (SQ). Also, the observations and predictions from the recent 2 days are effective. These results are consistent with the findings from previous work (e.g., (Cai et al., 2014a; Jiang et al., 2014b)) that the predicted popularity dominates the QAC rankings. However, other signals also contribute many useful features, e.g., semantic query similarity represented by temporal relation and term pairwise occurrence frequency, e.g., $f_{TemRelSum}$ and f_{LLRSum}. Two particularly interesting observations from Table 5.12 are that: (1) word2vec features are ranked very high, suggesting that the developed semantic relatedness among term pairs inside a query does indeed help to generate appropriate queries; (2) the majority of important features use maximization rather than summation of values.

Table 5.12: Ten most important features by χ^2 test on the AOL and MSN datasets; $PI(q)$ and $SQ(q)$ are placeholders for pseudo-identical queries and super queries of query q, respectively.

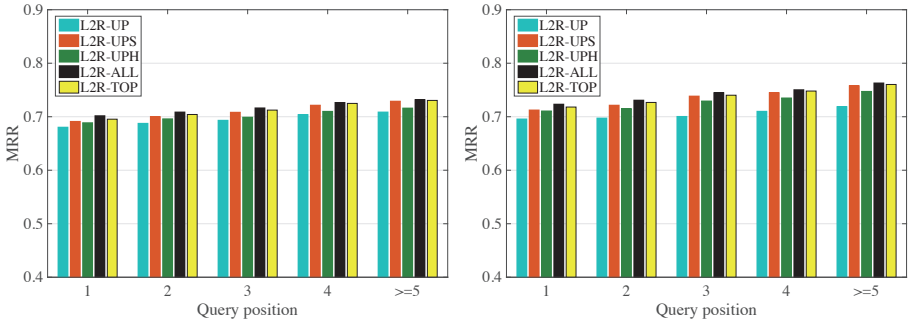
Rank	AOL	MSN
1	$f_{W2vLogMax}$	$f_{W2vLogMax}$
2	$f_{TemRelSum}$	$fre(PI(q),2)_{tre\_max}$
3	$fre(PI(q),2)_{tre\_max}$	$f_{W2vGooMax}$
4	$f_{W2vGooMax}$	$fre(PI(q),2)_{obs\_max}$
5	$fre(PI(q),4)_{obs\_max}$	$f_{TemRelSum}$
6	$f_{TemRelMax}$	$fre(PI(q),4)_{tre\_max}$
7	f_{cof_sum}	$fre(PI(q),1)_{tre\_max}$
8	$fre(PI(q),2)_{obs\_max}$	$f_{W2vLogSum}$
9	$f_{W2vLogSum}$	f_{LLRSum}
10	$fre(SQ(q),2)_{tre\_sum}$	$fre(SQ(q),4)_{tre\_max}$

To verify the effectiveness of features deemed to be important for QAC, we create a model called L2R-TOP that extends L2R-UP with the features in Table 5.12 (on the corresponding datasets). The results in terms of MRR scores are listed in Table 5.9, column 7. We see that (1) compared to features selected from a sole source into L2R-UP, i.e., semantic relatedness or homologous queries, the important features according to Table 5.12 boost the performance; L2R-TOP receives higher MRR scores than L2R-UPS and L2R-UPH; (2) L2R-ALL invariably performs the best among all models, suggesting that L2R-based models not only learn from the important features, but also from the less important ones. Overall, L2R-TOP produces competitive results, implying that its 63 features (the ten most important plus 53 from L2R-UP) are highly informative for producing high quality QAC rankings.

5.3.6 Impact of Query Position

Previous work mainly focuses on the last query in a session for QAC evaluation (Cai et al., 2014b; Jiang et al., 2014b). Instead, we implement tests on all queries in a session, which helps us to examine the performance of our L2R-based QAC models at various query positions in a search session. We plot the results in terms of MRR in Figure 5.5 for all prefixes at each specific query position in a session, tested on the AOL and MSN datasets, respectively.

We can see from Figure 5.5 that: (1) for all L2R-based QAC models, the QAC performance in terms of MRR is improved when the user continues querying in a session because the MRR scores are increased as the query position changes from the beginning to the end of a session; (2) among these models, the performance of L2R-UP seems to be less sensitive to the query position than other models, especially on the MSN dataset, as the MRR scores of L2R-UP are relatively stable; (3) generally, the L2R-ALL model invariably performs best among these models at each specific query position. These findings could be due to: (1) at the end of a search session, the information of user behaviors makes more sense for learning than at the beginning of a search session, which helps



(a) AOL **(b) MSN**

Figure 5.5: Performance of L2R-based QAC models in terms of MRR at various query positions, tested on the AOL and MSN datasets, respectively.

boost the QAC ranking performance; (2) semantic features are more reliably extracted at the end of a session rather than at the beginning, especially for features depending on the search context, such as, e.g., $f_{TemRelSum}$ in (5.8) and f_{cof_sum} in §5.1.3, which are important to those models, e.g., L2R-UPS, L2R-ALL and L2R-TOP.

5.4 Conclusion

In this chapter we follow a supervised learning to rank approach to address the problem of ranking query auto completion (QAC) candidates. We develop new features of homologous queries (i.e., those with the same terms but different orders and those extending the initial query) and semantic relatedness of terms inside a query and of pairs of terms from a query candidate and from earlier queries in the same session. We have verified the effectiveness of our models on two public datasets, showing significant improvements over state-of-the-art QAC baselines. Our analysis reveals that features of semantic relatedness and homologous queries are important and do indeed help boost QAC performance.

As to future work, we will have a closer look at the top N candidates returned by popularity based candidate ranking (MPC) for $N > 10$: how much can we gain from good candidates that were ranked low by MPC? Additionally, we want to study efficiency aspects of our approaches: parallel processing is likely to boost the efficiency of our models on feature extraction, and the addition of more, potentially expensive ways of generating homologous queries or semantic features could produce better QAC rankings. Finally, we aim to apply our approach to larger datasets than we considered in this paper, especially datasets that cover longer periods of time than AOL and MSN, as we believe that QAC can benefit from periodicity-based features.

In Chapter 4, we focused on time-sensitivity and user-specific context for query auto completion. In this chapter, we mainly focused on exploring information from homologous queries and from semantically related terms for query auto completion. In the next chapter, we will investigate whether diversifying the list of query completions can boost the performance of query auto completion as well as improve the satisfaction of users.

6

Diversifying Query Auto Completion

So far, we have explored the information of time-sensitivity and of user-specificity for query auto completion (QAC) in Chapter 4, and focused on learning from homologous queries and semantic relatedness for query auto completion in Chapter 5. In both chapters, we have seen that previous work on query auto completion, including our own proposed models, mainly centers around returning completions that are most likely intended by the user while ignoring the possible redundancy among the query completions in the list. Thus, semantically related queries matching the input prefix are often returned together. This may push valuable suggestions out of the list, given that only a limited number of candidates can be shown to the user, and hence, this may result in a less than optimal search experience. Therefore, without further information to disambiguate the user's query aspect, the search engine needs to focus on how best to produce a list of *relevant* and *diversified* query completions that can cover different interpretations.

Intuitively, a sensible QAC approach should maximize the satisfaction of the population of users typing the prefix, which involves a trade-off between presenting more possible query completions of the "correct" query aspect and having diverse query completions in the top positions of the list of query completions for a given prefix, i.e., returning the most probable queries early and removing redundant query candidates as well. By doing so, the chance that any user typing the same prefix will find at least one satisfactory query candidate for their particular information need is maximized. Hence, it is important to capture the user's query aspect and reduce the redundancy of query completions.

Let us illustrate the meaning of query completion redundancy, which refers to the situation where some auto-completed queries are of equivalent meaning to preceding queries in the list of query completions, describing almost the same query aspect. Table 6.1 contains a search session from the well-known AOL query log (Pass et al., 2006). For the sake of the example, let us assume that we have not yet seen the last query ("*sony*" in row 6) and that it is in fact the initial segment (prefix) "*so*" of this query for which we want to recommend query completions. A regular baseline (Bar-Yossef and Kraus, 2011) based on the most popular query is likely to rank the query completions as a list shown in row 8. But if we look in the list (e.g., query completions at rank 1, 6 and 9), we see that the queries "*southwest airlines*," "*southwestairlines*" and "*southwest airline*" are all returned. Clearly, these three candidates are semantically closely related to each other and probably express an identical query aspect, which can be easily confirmed by con-

Table 6.1: A session example from the AOL dataset consisting of five queries (rows 2–6), and a ranked list of top ten query completions (row 8), separated by ";" and returned by MPC (Bar-Yossef and Kraus, 2011) after typing the prefix "*so*" of the last query "*sony*".

1	SessionID	UserID	Query	Time
2	419	1020	compaq	20060315, 14:18:42
3	419	1020	hewlit packard	20060315, 14:26:58
4	419	1020	toshiba	20060315, 14:32:31
5	419	1020	averatec	20060315, 14:35:39
6	419	1020	sony	20060315, 14:38:15
7	A ranked list of query completions for the prefix "so"			
8	southwest airlines; southwest; song lyrics; social security; sopranos; southwestairlines; sony; sofia laiti; southwest airlines; social security administration			

sidering the overlap of the search engine result pages (SERPs) produced for these three queries. We hypothesize that in this example, the latter two candidates, i.e., "*southwest-airlines*" and "*southwest airline*," in this list of query completions (row 8) are redundant completions for the prefix "*so*." Thus, to improve the user's search satisfaction, they should be removed from the list, especially when only few query completions can be returned. By doing so, the QAC performance in this case, typically measured by Mean Reciprocal Rank (MRR), can be improved given that the final submitted query is "*sony*." Moreover, in this case, if the final query submission is "*social security administration*," the MRR score obviously can be further increased as more redundant candidates before "*social security administration*" in the list of query completions are cleared.

After removing redundant queries from the list of query completions, more query completions can be included, which increases the probability of matching the intended query of the user. Thus, in this chapter, we consider the task of *diversifying query auto completion* (D-QAC), which aims to return the correct query early in the QAC ranking list and to reduce the redundancy among the query completions. In particular, we propose a series of greedy query selection (GQS) models, i.e., GQS_{MPC+AQ}, GQS_{MSR+AQ}, GQS_{MPC+LQ} and GQS_{MSR+LQ}, corresponding to a GQS model that first selects the most popular completion and use all previous queries in session as search context, that first selects the most similar completion and use all previous queries in session as search context, that first selects the most popular completion and use only the last preceding query in session as search context and that first selects the most similar completion and use only the last preceding query in the session as search context, respectively. We identify a query's aspects by categorizing its clicked URLs using the ODP (Open Directory Project) taxonomy,[1] a topical hierarchy structure for URL categorization.

In practice, our GQS model faces two main challenges. One relates to a query cold-start problem. When ranking the query candidates in the testing phase, we may not know all aspects of a query candidate from the logs in the training period. The other

[1] `http://www.dmoz.org`

problem is a sparseness problem: we only have limited aspect information about every query. For the query cold-start problem, we propose a solution by which an unlabelled query can be assigned the same aspects as its semantically most closely related query that has been labelled by ODP in the training period. For the sparseness problem, we apply a Bayesian probabilistic matrix factorization approach to derive a distribution of query over all aspects.

We answer the following questions for this problem:

RQ12 Do our greedy query selection (GQS) models beat the baselines for diversifying query auto completion task in terms of metrics for QAC ranking (e.g., MRR) and for diversification (e.g., α-nDCG)?

RQ13 How does the choice of selecting the first query to be included in the QAC result list impact the performance in diversified query auto completion of our GQS model?

RQ14 What is the impact on diversified query auto completion performance of our GQS model of the choice of search context, i.e., choosing all previous queries in a session or only the last preceding query?

RQ15 What is the relative D-QAC performance of our QAC models when evaluated using a side-by-side comparison?

RQ16 What is the sensitivity of our GQS model? In particular, how is the performance of our GQS model influenced by, e.g., the number of returned query completions, namely a cutoff N, the number of latent features used in BPMF k_f and a trade-off λ controlling the contribution of search popularity and search context when modeling the closeness of query completion to search intent?

We answer these questions by changing the scenario of selecting the first query completion to the final list and of collecting session context for inferring search intents. Our experimental results for the GQS models, results obtained on two large-scale real-world query logs, show that our proposal can outperform a competitive state-of-the-art baseline in terms of well-known metrics used in QAC and diversification, e.g., MRR and α-nDCG, respectively. We also conduct a side-by-side comparison to assess the diversity of QAC suggestions.

Our contributions in this chapter can be summarized as:

1. We propose the task of diversifying query auto completion (D-QAC) that aims to return the user's intended query early in a list of query completions and, simultaneously, to reduce the redundancy of the QAC list. To the best of our knowledge, there is no published work on D-QAC.
2. We propose a greedy query selection (GQS) approach for D-QAC that captures the query aspect not only from the current search popularity but also from the search context in session.
3. We study a query cold-start problem, where we do not have any aspect information about a query from the logs in the training period. For such unknown queries, we assign aspect labels from its semantically most closely related query for which aspect information has been found during the training period.
4. We analyze the effectiveness of our GQS model and find that it significantly outperforms state-of-the-art baselines for D-QAC in terms of MRR and α-nDCG. Generally, against the best baseline, GQS achieves an improvement of around 2.3% and 5.6% in terms of MRR and α-nDCG, respectively.

Table 6.2: Main notation used in the chapter.

Notation	Description
T	number of queries in a session
q_t	the t-th ($t = 1, 2, \ldots, T$) query in a session
p	prefix of the last query q_T in a session
R_R	a list of query completions for prefix p returned to the user
R_I	an initial QAC ranking list matching prefix p
k_I	number of query completions in R_I, i.e., $\|R_I\|$
a	aspect
$q(i)$	the probability of relevance of query q to the i-th aspect
C_S	search context, i.e., sequence of queries preceding q_T: $\{q_1, q_2, \ldots, q_{T-1}\}$
q_c	query candidate in a QAC list
λ	trade-off between search popularity and previous search context
$f(q)$	frequency of query q
θ	decay factor
q_s	selected query in R_R
ω_t	normalized decay brought by temporal interval
$TD(q_t)$	time interval between q_t and q_T
N	number of query completions finally returned to a user, i.e., a cutoff
Q	set of unique queries
Q_L	set of labelled queries $\subseteq Q$
$\mathbb{A}$	set of unique aspects
N_q	number of unique queries, i.e., $\|Q\|$
M_a	number of unique aspects, i.e., $\|\mathbb{A}\|$
k_f	number of latent features used in BPMF

The remainder of this chapter is organized as follows. The D-QAC problem and our proposed solution are described in §6.1. Section 6.2 presents our experimental setup. In §6.3, we report our results. Finally, we conclude in §6.4, where we also suggest future research directions.

6.1 Approaches

In this section, we formally introduce the problem of diversifying query auto completion (D-QAC) in §6.1.1, describe our greedy query selection model to deal with D-QAC in §6.1.2, and derive query distributions over aspects in §6.1.3.

6.1.1 The D-QAC Problem

Before introducing our method for D-QAC, we first recall the main notation used in this chapter in Table 6.2 and then state the problem of D-QAC. As the search engine only returns the top N query completions to its users, the objective of D-QAC is to maximize the probability that the average user finds at least one useful query completion within the top N candidates.

Assume that the following are given:

- a prefix p of the last query q_T in a session consisting of T queries $\{q_1, q_2, \ldots, q_T\}$;
- an initial list of query completions R_I produced for this prefix p with length $|R_I| = k_I$;
- the probability of relevance $P(Rel \mid a, p, C_S)$ of query aspect a for prefix p given the search context C_S consisting of a sequence of preceding queries before q_T, i.e., $\{q_1, q_2, \ldots, q_{T-1}\}$;
- and a satisfaction value $P_s(Rel \mid q_c, p, a, C_S)$, i.e., a probability of query completion q_c matching the query aspect a given the search context C_S.

First, we start with a simplified objective of the *diversified query auto completion* (D-QAC) problem, which aims to satisfy the average user who enters the prefix p by finding at least one acceptable query completion among the top N query completions returned, given his search context C_S, where $R_R \subseteq R_I$ with $|R_R| = N$, such that $N \leq k_I$. This is achieved by maximizing

$$P(R_R \mid p, C_S) = P(Rel \mid p, C_S) \left(1 - \prod_{q_c \in R_R} (1 - P_s(Rel \mid q_c, p, C_S))\right). \quad (6.1)$$

Let us illustrate this objective and see how it formalizes our intuition. Note that $P_s(Rel \mid q_c, p, C_S)$ can be interpreted as the probability that a query completion q_c satisfies a user who enters the prefix p given the search context C_S. Then $(1 - P_s(Rel \mid q_c, p, C_S))$ indicates the probability that q_c fails to satisfy the user. Therefore, the probability that all selected query completions fail to satisfy the user equals its product under the query independence assumption. One minus that product is the probability that at least one query completion satisfies the user. Finally, the score, weighted by $P(Rel \mid p, C_S)$, denotes the probability that the set of query completions R_R satisfies the average user. Then, we break the objective in (6.1) down to the aspect level as

$$P(R_R \mid p, C_S) = \sum_a P(Rel \mid a, p, C_S) \left(1 - \prod_{q_c \in R_R} (1 - P_v(Rel \mid q_c, p, a, C_S))\right), \quad (6.2)$$

where a is a given aspect and summing over all aspects weighted by $P(Rel \mid a, p, C_S)$ denotes the probability that the set of query candidates R_R satisfies the average user who enters prefix p at the aspect level.

This D-QAC framework promotes diverse rankings of query completions by penalizing redundancy at every rank in the list of query completions. It does so by greedily selecting query completions from $R_I \backslash R_R$ into R_R. At each step, it selects one candidate that is most different from those previously selected in R_R (thus minimizing redundancy), while still relevant to the query aspect. This can be achieved by iteratively filling R_R with one query $q^\star \in R_I \backslash R_R$ each time until $|R_R| = N$:

$$q^\star \leftarrow \operatorname*{argmax}_{q_c \in R_I \backslash R_R} \sum_a P(Rel \mid q_c, p, a, C_S) \prod_{q_s \in R_R} (1 - P(Rel \mid a, p, q_s, C_S)), \quad (6.3)$$

where $P(Rel \mid q_c, p, a, C_S)$ denotes the probability that, for prefix p, candidate q_c meets the query aspect a given the search context C_S, and $P(Rel \mid a, p, q_s, C_S)$ indicates the

conditional probability that the selected query q_s for prefix p matches the aspect a given the search context C_S. Thus,

$$\prod_{q_s \in R_R} (1 - P(Rel \mid a, p, q_s, C_S))$$

denotes the probability that all previously selected queries $q_s \in R_R$ fail to satisfy the user, and the product

$$P(Rel \mid q_c, p, a, C_S) \prod_{q_s \in R_R} (1 - P(Rel \mid a, p, q_s, C_S))$$

indicates the probability that none of the selected queries in R_R satisfy the query aspect a but finally q_c does. Finally, the query completion $q_c \in R_I \backslash R_R$ with the maximal probability satisfying the query aspect is chosen for inclusion in the list R_R.

6.1.2 Greedy Query Selection for D-QAC

In this section, we propose our Greedy Query Selection (GQS) model to deal with D-QAC. In this model, we assume that the probability in (6.3) that a query candidate q_c meets the query aspect $P(Rel \mid q_c, p, a, C_S)$ can be expressed either by the current search popularity or implicitly by the closeness to previous queries in the session context C_S, with a trade-off λ ($0 \leq \lambda \leq 1$) controlling the contributions from these two parts. Thus

$$\begin{aligned} P(Rel \mid q_c, p, a, C_S) &= \lambda P(q_c \mid p) + (1 - \lambda) P(Rel \mid q_c, a, C_S) \\ &= \lambda P(q_c \mid p) + (1 - \lambda) P(Rel \mid q_c, a, q_1, q_2, \ldots, q_{T-1}) \\ &= \lambda P(q_c \mid p) + (1 - \lambda) \prod_{q_t \in C_S} P(Rel \mid q_c, a, q_t), \end{aligned} \tag{6.4}$$

where the query aspect expressed by search popularity $P(q_c \mid p)$ can be directly estimated by

$$P(q_c \mid p) = \frac{f(q_c)}{\sum_{q \in R_I} f(q)}, \tag{6.5}$$

where $f(q)$ indicates the frequency of query $q \in R_I$. Before combining these two probability scores, they should be normalized. Obviously, the query aspect is dominated solely by the search popularity if $\lambda = 1$ or by the search context if $\lambda = 0$. The probability $P(Rel \mid q_c, a, q_t)$ in (6.4) can be simply estimated by the normalized distance between q_c and q_p given a specific aspect a, weighted by the temporal intervals:

$$P(Rel \mid q_c, a, q_t) = \omega_t \times \left(1 - \frac{|q_c(a) - q_t(a)|}{dis(q_c, q_t)}\right), \tag{6.6}$$

where $dis(q_c, q_t)$ returns the 2-norm distance between q_c and q_t and ω_t is a normalized decay factor brought by the temporal interval between q_t and q_c (namely q_T) to make $\sum \omega_t = 1$, as we argue that temporally close queries in a search session are apt to share common query aspects. Actually, any other similarity function can be used, including,

e.g., cosine similarity. However, for computing the distance at a specific aspect, we do need to normalize the score at this specific aspect dimension. Different similarity functions will return similar results. Note that, in our model, all queries are represented by a vector, where each entry in this vector indicates the relevance of the query to a particular aspect a. Specifically, $q_c(i)$ denotes the probability of relevance of query q_c represented by a vector to its i-th aspect. We compute ω_t as $\omega_t \leftarrow norm(\theta^{TD(q_t)-1})$, where θ is a decay factor and $TD(q_t)$ refers to the time interval, e.g., $TD(q_t) = 1$ for the last query q_{T-1} in the context C_S. The query, e.g., q_c or q_t, can be represented by a probability distribution over aspects returned by Bayesian probabilistic matrix factorization, which is to be discussed in Section 6.1.3. Hence these probabilities can be computed offline before ranking.

Next, the probability $P(Rel \mid a, p, q_s, C_S)$ in (6.3) indicates to what degree the selected query completion $q_s \in R_R$ meets the query aspect, which can be learnt from the search logs. We simplify $P(Rel \mid a, p, q_s, C_S)$ in (6.3) as:

$$P(Rel \mid a, p, q_s, C_S) = P(Rel \mid a, q_s, C_S), \tag{6.7}$$

indicating the probability that a query candidate matches the query aspect is dominated by the closeness to the aspect of preceding queries in the session. Finally, based on the aforementioned query independency assumption, we have

$$P(Rel \mid a, p, q_s, C_S) = P(Rel \mid a, q_s, C_S) \propto \prod_{q_t \in C_S} P(Rel \mid q_s, a, q_t), \tag{6.8}$$

where $P(Rel \mid q_s, a, q_t)$ can be derived in the same way as $P(Rel \mid q_c, a, q_t)$ in (6.4). By doing so, we can gradually assign one query candidate into the list R_R at a time until reaching the list length $|R_R|$. The details of our query selection method can be found in Algorithm 5. We first calculate the semantic similarity of query candidates to the search context (row 3), and inject the most semantically similar query into R_R, shown in row 6, Algorithm 5, then score the remaining candidates in R_I by measuring how are they close to the query aspects and at the same time how are they diverse to the selected query in R_I (see row 10), and finally select the optimal candidate to R_I by row 12 and 13. We iteratively select one candidate at a time from the remaining list until $R_R = N$.

Clearly, as shown in Algorithm 5, first of all, we should initialize R_R. We consider two options. The first option starts with $R_R \leftarrow q^*$, where q^* is the most popular completion in R_I at the time of querying because popular queries normally receive high attentions. We write GQS$_{MPC}$ to denote this variant of the GQS model. Another option initializes R_R with the most semantically related query to the previous queries in the search context, where q^* can be directly returned by using word2vec (Mikolov et al., 2013b) because queries in the same session normally express closely similar aspects. We write GQS$_{MSR}$ to denote the variant of our GQS model that starts with the semantically most related query.

Finally, to get these probabilities used in our GQS model, e.g., $P(Rel \mid q_c, a, q_t)$ in (6.4) and $P(Rel \mid q_s, a, q_t)$ in (6.8), we should know the query distribution over all aspects, which is returned by an algorithm based on BPMF (Bayesian probabilistic matrix factorization), thereby overcoming the sparseness problem of not knowing the direct relationship between a query and an aspect, see §6.1.3. However, BPMF needs

Algorithm 5 GQS for D-QAC.

Input: Prefix p, an initial QAC list R_I, size of returned QAC list: N, search context C_S
Output: A reranked QAC list R_R;

$R_R = \emptyset$
for each candidate $q_c \in R_I$ **do**
  $FirstQuery(q_c) \leftarrow Similarity(q_c, C_S)$; %%% Alternatively, $MPC(q_c)$
end for
$q^\star \leftarrow \arg\max_{q_c \in R_I} FirstQuery(q_c)$
$R_R \leftarrow R_R \cup \{q^\star\}$
$R_I \leftarrow R_I \backslash \{q^\star\}$
for $|R_R| \leq N$ **do**
  for $q_c \in R_I$ **do**
    $s(q_c) \leftarrow \sum_i P(Rel \mid q_c, p, a, C_S) \prod_{q_s \in R_R} (1 - P(Rel \mid a, p, q_s, C_S))$
  end for
  $q^\star \leftarrow \arg\max_{q_c} s(q_c)$
  $R_R \leftarrow R_R \cup \{q^\star\}$
  $R_I \leftarrow R_I \backslash \{q^\star\}$
end for
Return R_R

to know at least one aspect label of each query while it indeed happens that we may not know any aspect information about a query. To address this, we use the scenario proposed in Algorithm 6 by finding a labelled target query that is the semantically most closely related query to the unlabelled query.

6.1.3 Generating Query Distribution Over Aspects

In this section, we discuss how to generate a query distribution over aspects. We use Bayesian probabilistic matrix factorization to overcome the sparseness problem of not knowing direct relationships between a query and an aspect using ODP. Before detailing our Bayesian probabilistic matrix factorization-based approach, we address the query cold-start problem, i.e., not knowing any aspect information about a query using ODP categorization from the training period. We assign the aspect labels from its semantically most related query in the labelled query set, because semantically related queries (or words), which often express similar search aspects, have been either directly suggested to the user or internally used by the search engine to improve the search quality (Bollegala et al., 2007; Chien and Immorlica, 2005).

More precisely, given an unlabelled query q and a set of labelled queries Q_L, we return a labelled query $q_o \in Q_L$ for q as:

$$q_o \leftarrow \operatorname*{argmax}_{q_l \in Q_L} \cos(q, q_l) = \operatorname*{argmax}_{q_l \in Q_L} \frac{1}{W} \sum_{w_k \in q} \sum_{w_j \in q_l} \cos(w_k, w_j). \tag{6.9}$$

We take the cosine similarity between two queries, represented by the average of the cosine similarity between two sets of normalized word vectors, excluding the stop words.

Algorithm 6 Dealing with query cold-start problem.

Input: An unlabelled query q;
a set of labelled queries Q_L with their labels L
Output: Labels of q: $l(q)$;
1: **for** each query $q_l \in Q_L$ **do**
2: $score(q_l) = \cos(q, q_l)$
3: **end for**
4: $q_o \leftarrow \text{argmax}_{q_l \in Q_L}\, score(q_l)$
5: $l(q) \leftarrow l(q_o) \in L$
6: **Return** $l(q)$ to q

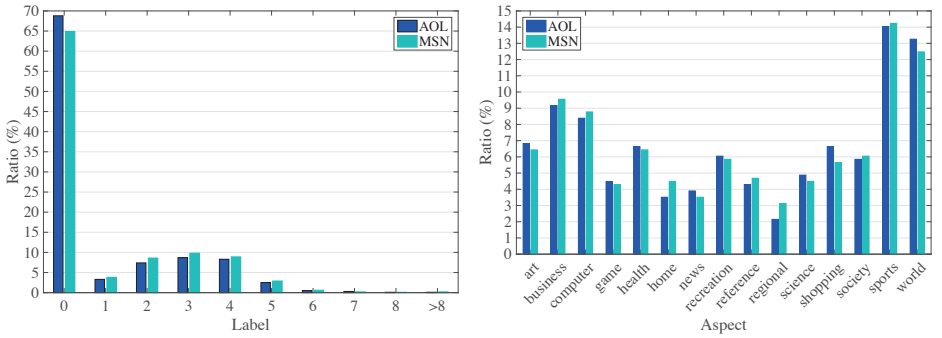
The word vector representation can be directly returned by word2vec (Mikolov et al., 2013a,b) learnt from the query logs. The details are shown in Algorithm 6, where we first score each labelled query in Q_L by its cosine similarity to the unlabelled input query in row 2, and then select the most similar query (row 4), from which we obtain the aspect labels that are finally assigned to the input query as aspect labels (row 5). Using Algorithm 6, all the queries in our datasets can be categorized by ODP. After that, BPMF can be applied to derive the query distribution over all aspects in order to calculate the probabilities used in Algorithm 5 to rerank query completions. However, queries are usually short. E.g., the majority in the AOL and MSN logs that we use in this chapter consists of fewer than three words; see Figure 6.2b below. Hence, it makes sense to use the clicked documents rather than the query itself to identify query aspects, which is commonly used in search result diversification. We thus build a large query-aspect matrix $QC_{N_q \times M_a}$ using ODP, with N_q unique queries and M_a unique aspects, as we will explain.

In the diversity task of TREC,[2] the ground truth is generated by humans, both for relevance and for aspects. In our setting of diversified query auto completion, we first train our model by proposing a new approach for inferring multi-aspect relevance for a query from clickthrough data in the log using aspect information from the open directory project, ODP.[3] The clickthrough data is produced from the search behaviors of real searchers and has been proved effective for labelling the relevance of a document to a query (Joachims, 2002). More specifically, our methodology consists of two major steps. The first step involves extracting the clickthrough data from the search log. By doing so, we obtain a list of all clicked URLs for each unique query. The second step involves categorizing these URLs using ODP. After that, we infer the aspects of a query by aggregating all aspects from its clicked URLs. Let us make this more precise.

Definition 6.1.1 (Multi-aspect relevance) Let a query be given. Given an aspect set that has m pertinent aspects, i.e., an aspect relevance label is independent of other aspect relevance labels, the multi-aspect relevance of a query is an m-dimensional vector with each entry corresponding to a relevance label for an aspect given the query.

Hence, each entry in a multi-aspect relevance vector corresponds to an aspect relevance

[2] `http://trec.nist.gov/tracks.html`
[3] DMOZ – the open directory proejct, `http://www.dmoz.org`.



(a) The ratio of relevance labels of queries at two-level aspects in the AOL and MSN datasets.

(b) The ratio of queries at one-level aspects in the AOL and MSN datasets.

Figure 6.1: Distributions of labels (left) and aspects (right) in the AOL and MSN datasets, respectively.

label. This relevance label can be mapped to a numerical value $n_e(q, url, a)$ as:

$$n_e(q, url, a) = \sum_{url \in U(q)} J(url, a) \times f(q, url), \tag{6.10}$$

where $U(q)$ contains all clicked URLs of a query q, the indicator $J(url, a) = 1$ if the clicked url is categorized by aspect a and $J(url, a) = 0$ if not, and $f(q, url)$ indicates the number of clicks on URL url after submitting query q. We use ODP to categorize the clicked URLs. In practice, following (Chapelle et al., 2009), we split the aspect relevance judgments into a 5-grade scale set, like $\{perfect, excellent, good, fair, bad\}$ by $n_e(q, d, a) \leftarrow \min(n_e(q, d, a), 4)$. After checking the distribution of labels (see Figure 6.1a) and aspects[4] (see Figure 6.1b), we find the relevance labels, e.g., 2, 3 and 4, account for the majority of non-zero labels. In this manner, we generate our ground truth for the relevance of queries to aspects.

To calculate the probabilities mentioned in Section 6.1.2, i.e., $P(q_c|p, a, C_S)$ in (6.4) and $P(a|p, q_s, C_S)$ in (6.7), we should replace the zeros in the original query-aspect matrix $QC_{N_q \times M_a}$ for the cases that no direct relationships between query q and aspect a are inferred using ODP in the training period. We use Bayesian Probabilistic Matrix Factorization (BPMF) (Salakhutdinov and Mnih, 2008a) to derive the distribution of queries over aspects. BPMF can be directly applied to the original query-aspect matrix $QC_{N_q \times M_a}$, returning an approximation matrix that assigns a non-zero value to each entry in the original matrix to overcome the problem of sparseness and zero-probabilities. By doing so, the original query-aspect matrix $QC_{N_q \times M_a}$ is approximated by:

$$QC^*_{approx} = Q^*_{N_q \times k_f} \times {C^*_{M_a \times k_f}}^\top, \tag{6.11}$$

where $Q^*_{N_q \times k_f}$ and $C^*_{M_a \times k_f}$ represent the query- and aspect-specific latent feature matrix, and N_q, M_a and k_f indicate the number of queries, aspects and latent features,

[4]To save space, we only plot the distribution of level-one aspects, in total 15 aspects.

respectively. Like (Cai et al., 2014a), we compute the value $QC^*_{approx}(i,j)$ of query i at aspect j in matrix QC^*_{approx} by marginalizing over the model parameters and the hyperparameters:

$$p(QC^*_{approx}(i,j)|QC_{N_q \times M_a}, \Theta_0) = \tag{6.12}$$
$$= \int\int p(QC^*_{approx}(i,j)|Q_i, C_j) p(Q, C|QC_{N_q \times M_a}, \Theta_Q, \Theta_C) p(\Theta_Q, \Theta_C|\Theta_0)$$
$$d\{Q, C\} d\{\Theta_Q, \Theta_C\},$$

where $\Theta_Q = \{\mu_Q, \Sigma_Q\}$ and $\Theta_C = \{\mu_C, \Sigma_C\}$ are hyperparameters of query set Q consisting of all unique queries and of aspect set $\mathbb{A}$ consisting of all unique aspects, respectively. The prior distributions over the query and aspect feature vectors are assumed to be Gaussian, and $\Theta_0 = \{\mu_0, \Sigma_0, W_0\}$ is a Wishart distribution hyperparameter with $\Sigma_0 \times \Sigma_0$ scale matrix W_0. The intuition behind this approximation is that the relevance of a query to aspects is determined by a small number of unobserved hyperparameters. This means that taking a Bayesian approach to the prediction problem involves integrating the model hyperparameters. In addition, the use of Markov chain Monte Carlo (MCMC) methods (Neal, 1993) for approximating relevance comes from finding only point estimates of model hyperparameters instead of inferring the full posterior distribution over them, which results in a significant increase in predictive accuracy (Salakhutdinov and Mnih, 2008a).

BPMF introduces priors for the hyperparameters, which allows the model complexity to be controlled based on the training data (Salakhutdinov and Mnih, 2008b). When the prior is Gaussian, the hyperparameters can be updated by performing a single EM step (Dempster et al., 1977), which scales linearly with the number of observations without significantly affecting the time to train the model. The details of BPMF can be found in (Salakhutdinov and Mnih, 2008a). As some of the values in the matrix QC^*_{approx} generated by BPMF are negative, we normalize QC^*_{approx} to guarantee $QC^*_{approx}(i,j) \in (0,1)$. After normalizing, distributions of queries over aspects can be produced.

6.2 Experiments

Below, §6.2.1 provides an overview of the models for D-QAC studied in this chapter; §6.2.2 describes the datasets; §6.2.3 details the design of a side-by-side experiment, and we specify our settings and parameters in §6.2.4.

6.2.1 Model Summary

We list all the models to be discussed in Table 6.3. There are three state-of-the-art baselines and four flavors of approaches that we introduce in this chapter: greedy query selection (GQS, Algorithm 5) with two notions of context (AQ, all preceding queries vs. LQ, only the last query) and two notions of starting query (MPC, most popular query, vs. MSR, semantically most closely related query).

In addition to the MPC model (Bar-Yossef and Kraus, 2011), which is referred to as QD-MPC in this chapter and has been introduced as a baseline approach in Chapter 4 and 5, we consider three more QAC baselines:

Table 6.3: An overview of models discussed in the chapter.

Model	Description	Source
QD-MPC	A QAC ranking approach, which ranks query completions according to their current popularity collected from the query logs.	(Bar-Yossef and Kraus, 2011)
QD-CON	A context-based query ranking approach, which reranks query completions (returned by MPC) by a hybrid score considering the query popularity and the similarity to search context in current session.	(Bar-Yossef and Kraus, 2011)
QD-QCR	A diversification-oriented query ranking approach, which reranks query completions (returned by MPC) by selecting queries from distinct query clusters.	(He et al., 2011)
QD-MMR	A diversification-oriented query ranking approach, which ranks query completions according to both their popularity and the dissimilarity between a query candidate to be selected and those previously selected.	(Carbonell and Goldstein, 1998)
GQS_{MPC+AQ}	Greedy query selection approach starting with the most popular query and taking all preceding queries in session as context.	This chapter
GQS_{MPC+LQ}	Greedy query selection approach starting with the most popular query and taking the last preceding query in session as context.	This chapter
GQS_{MSR+AQ}	Greedy query selection approach starting with the most semantically related query to the preceding queries in session and taking all preceding queries as context.	This chapter
GQS_{MSR+LQ}	Greedy query selection approach starting with the most semantically related query to the preceding queries in session and taking the last preceding query in session as context.	This chapter

QD-MPC A popularity-based QAC approach, which ranks the query candidates by their frequency as computed from the counts in the preceding log (Bar-Yossef and Kraus, 2011);

QD-CON A context-based QAC approach,[5] where we rerank the query candidates by a hybrid score considering the query popularity and the similarity to the search context in current session (Bar-Yossef and Kraus, 2011);

QD-QCR A diversification-oriented query ranking approach based on query clusters (He et al., 2011), which reranks query completions returned by MPC via selecting query from distinct clusters; the K-means clustering algorithm is applied to cluster the queries with a fixed number of clusters, i.e. 5, as we evaluate the results produced by returning at least 5 query completions in our experiments;

QD-MMR An MMR-based (Carbonell and Goldstein, 1998) query ranking, which considers the candidate's popularity as well as its dissimilarity to those previously selected queries with setting a trade-off to 0.5; in particular, the dissimilarity is computed as:

$$dissim(q_c, R_R) \leftarrow \frac{1}{|R_R|} \sum_{q \in R_R} (1 - \cos(q, q_c)), \tag{6.13}$$

where q_c is a query candidate to be selected and q is a query which has been selected in the query set R_R. Both queries are represented by a vector returned by Bayesian probabilistic matrix factorization as detailed in §6.1.3.

6.2.2 Datasets

We use the processed AOL and MSN datasets that have been described in Chapter 3 and used in Chapter 5. A major difference between the datasets used in Chapter 5 and in this chapter is that, in Chapter 5, we set the prefix to be the first 1–5 character(s) of all queries in a session; however, in this chapter, for each session, the prefix is set to be the first 1–5 character(s) of the last query in the session. To get the training/test set, we remove the input prefixes whose ground truth is not included in the top 20 query completions returned by MPC, to guarantee that the candidate set contains the final query submission, following standard practice in the experimental design for QAC tasks; see, e.g., (Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013). We remove all test cases where the final submitted query cannot be categorized by ODP in the training phase, i.e., we cannot infer the query aspect of such queries, making it impossible to generate the ground truth.

Table 6.4 details the statistics of the datasets used. More than half (64.9%) of all unique queries in AOL can be categorized using ODP. For the MSN log we reach a higher ratio (66.2%). In the AOL log, if the user clicked on a search result of a query, only the domain portion of the URL in the clicked result is listed; however, in the MSN log, the full URL is recorded. For consistency with the way we process the AOL logs, we only keep the domains of clicked URLs in the MSN log when inferring query aspects via clickthrough data. Notice also that, compared to the average number of queries in a session in the AOL log ($\sim$3.3), the MSN users submit more queries per session ($\sim$5.5).

[5] We implement the BPMF process to generate a rich query representation over aspects to overcome the sparsity problem.

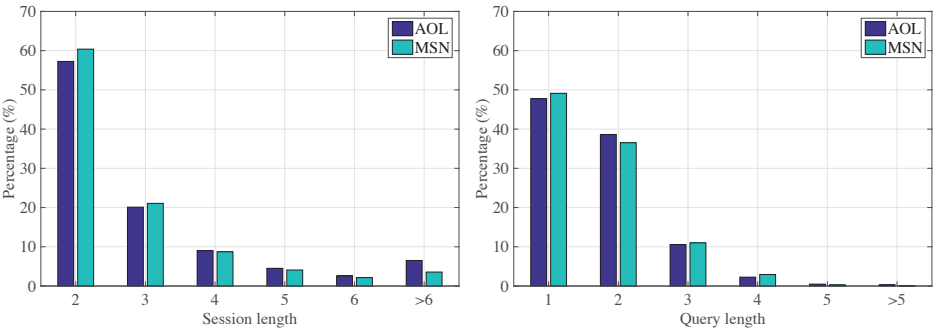
Table 6.4: Statistics of the AOL and MSN datasets used. The number of prefixes at various lengths are generated when top 20 query completions are returned by MPC, i.e., $k_I = 20$. The n in "# Prefix-n" indicates the length of the prefix in characters.

	AOL		MSN	
Variables	Training	Test	Training	Test
# Queries	3,808,083	1,571,346	3,784,925	1,402,308
# Unique queries	452,980	452,980	304,943	304,943
# Labelled Uniq Qs	294,363	294,363	201,872	201,872
# Unlabelled Uniq Qs	158,617	158,617	103,071	103,071
# Sessions	1,149,528	465,302	674,717	256,512
# Queries / session	3.31	3.38	5.60	5.46
# All prefixes	3,109,247	1,146,768	2,013,671	697,870
# Prefix-1	262,924	90,688	196,831	65,179
# Prefix-2	458,999	162,007	319,362	105,082
# Prefix-3	698,716	251,623	455,109	154,020
# Prefix-4	826,984	309,522	517,570	182,502
# Prefix-5	861,624	332,928	524,799	191,087

We take a closer look at the processed datasets to be able to report the ratio of queries at various lengths in words and of sessions at various lengths in queries in Figure 6.2. As shown in Figure 6.2a, for both datasets, nearly half of the sessions contain only two queries. The majority of sessions are short ($<$ 4queries), accounting for 77.3% in the AOL log and 81.2% in the MSN log. Long sessions ($>$ 6queries) are rare, accounting for only 6.5% in the AOL log and 3.5% in the MSN log. For the length of queries in words as shown in Figure 6.2b, more than 90% of the queries consist of at most three words, making it challenging to infer query aspects directly from queries.

6.2.3 Side-by-side Experiments

In addition to standard contrastive "bake-off" experiments, we use a second method for evaluating the diversity of QAC lists. Following (Chapelle et al., 2012; Thomas and Hawking, 2006; Vallet, 2011; Vallet and Castells, 2011), we show human judges in a lab setting two ranked lists of query completions for a given prefix and ask them which of the two is more diverse. This side-by-side evaluation experiment is performed using 50 master students in computer science. Each participant is given 50 different test prefix samples. For each test prefix, five lists of query candidates returned by QAC models, i.e., the baseline, GQS$_{MPC+AQ}$, GQS$_{MPC+LQ}$, GQS$_{MSR+AQ}$, and GQS$_{MSR+LQ}$, are presented in pairs to an individual participant, who is asked to indicate which query list is more diverse or whether there was a tie: GQS$_{MPC+LQ}$ vs. the baseline, GQS$_{MPC+LQ}$ vs. GQS$_{MPC+AQ}$, GQS$_{MSR+LQ}$ vs. the baseline and GQS$_{MSR+LQ}$ vs. GQS$_{MSR+AQ}$, respectively. During their assessments, participants were allowed to use a web search engine to help them decide. The selection of pairs of approaches for the side-by-side comparison is aligned with the comparisons for the standard contrastive "bake-off" experiments in §6.3.3. By doing so, we can examine the *agreement* on diversity preference between human judgments and the preferences inferred from the contrastive experiments.



(a) The ratio of sessions at various lengths in queries in the AOL and MSN dataset.

(b) The ratio of queries at various lengths in terms in the AOL and MSN dataset.

Figure 6.2: Distribution of session length in queries (left) and query length in terms (right) in the processed AOL and MSN datasets, respectively.

Table 6.5: Statistics of the side-by-side experiment.

Number of participants	50
Total number of prefixes assessed	2500
Number of prefixes assessed per prefix length (1, 2, 3, 4, 5)	500
Number QAC-candidates shown per prefix per model	10
Number of prefixes assessed per individual participant	50
Number of models compared	5
Number of pairs of QAC models judged	4

Table 6.5 summarizes the statistics of the side-by-side experiment.

Following (Chapelle et al., 2012), we use *agreement* to quantify to which extent the relative order of rankings obtained by computing the α-nDCG@10 scores of system-produced lists of QAC-candidates coincides with human preferences.

6.2.4 Parameters and Settings

Following (Bennett et al., 2012), we set the factor $\theta = 0.95$ in the decay function mentioned in §6.1.2. We first use one-third of the original test data for validation to optimize the free parameter λ controlling the contribution of query aspects signaled by the search popularity and the search context in (6.4), and then use the remaining two-thirds for the final test. We further discuss the influence of λ in §6.3.5. Similarly, the number of latent features used by Bayesian probabilistic matrix factorization in §6.1.3 is set to $k_f = 10$, which is followed by additional experiments and detailed discussions on the results generated when k_f is changing in §6.3.5.

For labeling query aspects, we use multiple relevance labels and proceed as follows. We first get a list of all clicked URLs of the remaining queries in the training period. Then we try to project each clicked URL for a query into a two-level ODP aspect based on the

links within each aspect[6] where the URLs have been categorized. Finally, given a query, we aggregate all aspects of its clicked URLs and use the aggregated aspect information to label the relevance of aspects for the query. Let us consider an example of the labelling process.

Example 6.2.1 Given a query q, we find that two URLs, e.g., d_1 and d_2, are clicked c_1 and c_2 times, respectively, according to the query log after q has been submitted. At the same time, based on the links within each ODP aspect, d_1 is labelled with aspects a_1 and a_2, and d_2 is labelled with aspects a_2 and a_3. Consequently, to query q we assign aspects a_1, a_2 and a_3, with counts c_1, $c_1 + c_2$ and c_2, respectively.

In total, we find 513 level-two topical aspects originated from 15 level-one topical aspects in our dataset based on this process.[7] These aspects are explicit. For instance, we can find aspects like "/arts/movies", "/shopping/crafts" and "/business/financial_services", where "/arts", "/shopping" and "/business" are level-one aspects, and "/movies", "/crafts" and "/financial_services" are their corresponding level-two aspects, respectively.

In previous research on QAC, it is often assumed that users are given a list of the top $N = 10$ (at most) query completions. This is a common setting used by many web search engines and in many publications (Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013). In our experiments, we first retrieve the top 20 query completions (at most) as determined by MPC and then return a final list of the top 10 (at most) candidates for evaluation after reranking the original QAC list by each specific model, i.e., $N = 10$ is initially used as a cutoff to test the diversified query auto completion performance. In addition, we examine the performance of our models at different cutoffs, i.e., at $N = 5$ and 20.

In practice, QAC methods should consider efficiency. Thus, the algorithm needs an efficient data structure, like a hash table, to support fast lookups for input prefix keys. Before testing, we generate an initial QAC ranking for each prefix offline by the MPC approach, and represent queries using vectors returned by the BPMF process. Then, for the diversification task, the main cost is on computing the similarities for reranking these query completions. All these preprocessing steps can be done offline.

6.3 Results and Discussion

In §6.3.1 we examine the performance of models for diversifying query auto completion in terms of MRR and α-nDCG@10, etc. We follow with a section discussing the scenario used in our GQS model for selecting the first query into the QAC list in §6.3.2. We examine the performance of our GQS model under different choices of the search context and zoom in on the performance at each prefix length in §6.3.3. We report on the outcomes of a side-by-side comparison on D-QAC performance in §6.3.4. Finally, §6.3.5 details the impact of the parameters used in our GQS model and provides an analysis of our GQS model under various settings.

[6] `http://www.dmoz.org/docs/en/rdf.html`

[7] As an aside, in total there are 16 level-one topical aspects in ODP; the aspect "*Kids and Teens*" was not found as an aspect in our dataset.

Table 6.6: Performance on the AOL and MSN logs based on the top 20 query completions initially returned by MPC, which are then re-ranked by the methods listed in the table. The results are reported at a cutoff of $N = 10$ for all prefixes. The values produced by the best baseline and the best performer in each column are underlined and boldfaced, respectively. Statistical significance of pairwise differences (GQS model vs. the best baseline) are determined by the t-test ($^{\blacktriangle}$/$^{\blacktriangledown}$ for α = .01, or $^{\triangle}$/$^{\triangledown}$ for α = .05).

Dataset	Method	MRR	ERR-IA@10	α-nDCG@10	NRBP	MAP-IA
AOL	QD-MPC	.5372	.3765	.6513	.3487	.2768
	QD-CON	.5391	.3782	.6526	.3488	.2783
	QD-QCR	.5393	.3791	.6538	.3491	.2794
	QD-MMR	.5377	.3783	.6530	.3490	.2785
	GQS$_{MPC+AQ}$	.5465	.3872$^{\triangle}$	.6681$^{\triangle}$	.3598$^{\triangle}$	.2864$^{\triangle}$
	GQS$_{MSR+AQ}$	.5509$^{\triangle}$	.3958$^{\blacktriangle}$	.6799$^{\triangle}$	.3632$^{\triangle}$	.2885$^{\triangle}$
	GQS$_{MPC+LQ}$	.5516$^{\triangle}$	.3965$^{\blacktriangle}$	.6852$^{\blacktriangle}$	.3645$^{\blacktriangle}$	.2898$^{\triangle}$
	GQS$_{MSR+LQ}$	**.5520**$^{\triangle}$	**.4007**$^{\blacktriangle}$	**.6901**$^{\blacktriangle}$	**.3679**$^{\blacktriangle}$	**.2907**$^{\blacktriangle}$
MSN	QD-MPC	.6158	.4184	.6562	.3891	.2546
	QD-CON	.6173	.4211	.6674	.4002	.2613
	QD-QCR	.6191	.4315	.6810	.4064	.2698
	QD-MMR	.6134	.4205	.6658	.3914	.2602
	GQS$_{MPC+AQ}$	.6285	.4417$^{\triangle}$	.6933	.4138	.2757$^{\triangle}$
	GQS$_{MSR+AQ}$	.6301	.4438$^{\triangle}$	.6994$^{\triangle}$	.4152$^{\triangle}$	.2771$^{\triangle}$
	GQS$_{MPC+LQ}$	.6307	.4452$^{\triangle}$	.7003$^{\triangle}$	.4174$^{\triangle}$	.2797$^{\triangle}$
	GQS$_{MSR+LQ}$	**.6324**$^{\triangle}$	**.4458**$^{\triangle}$	**.7025**$^{\triangle}$	**.4191**$^{\triangle}$	**.2794**$^{\triangle}$

6.3.1 D-QAC Performance of GQS

To answer our first research question, **RQ12**, we examine the D-QAC performance of all mentioned models in Table 6.3 and report the results in Table 6.6 for the AOL and MSN logs, respectively.

As shown in Table 6.6, QD-QCR achieves the best performance among the four baselines in terms of MRR and diversity metrics, e.g., α-nDCG@10. Hence, we only use QD-QCR as a baseline for comparisons with our proposed models in later experiments both on the AOL and MSN logs. In particular, for most test cases in these two datasets, all these baselines can return the final submitted queries early, at the top two positions, as evidenced by the fact that the MRR scores are larger than 0.5. In addition, the MRR scores of the baselines are close to each other. In particular, on the AOL log, QD-QCR achieves a competitive MRR score when compared against QD-CON but only achieves a minor MRR improvement ($< 0.5\%$) over QD-MMR and QD-MPC. QD-MMR displays a marginally better performance than QD-MPC in terms of MRR, indicating that for some cases QD-MMR is able to remove some redundant candidates in the original QAC list generated by the basic MPC approach. However, the MRR improvement is limited because QD-MMR partially relies on query popularity and it does not consider the in-session context when calculating the dissimilarity between queries. In contrast, on the MSN log, the baselines achieve somewhat higher MRR and diversity scores compared to those on the AOL log. This can be attributed to the difference in session length.

Compared to the sessions in the AOL log, the average length of sessions in the MSN log is longer and some repeated queries are found, which helps identify the query aspects of the last query in session and results in higher MRR and diversity scores.

In order to assess the performance of our GQS models, we compare their results against those of the selected baseline, i.e., QD-QCR. For the AOL log, as shown in Table 6.6, compared to the baseline, all of our four QGS models outperform it in terms of MRR and the diversity metrics. However, the improvements of our models are limited, starting from a relatively low level of 1.37% improvement achieved by GQS$_{MPC+AQ}$ to a peak improvement of 2.25% reported by GQS$_{MSR+LQ}$ in terms of MRR. In the middle, GQS$_{MPC+LQ}$ outperforms GQS$_{MSR+AQ}$, reaching an MRR improvement of 2.28% against the baseline while 2.15% made by GQS$_{MSR+AQ}$ against the baseline. All improvements of our GQS models, except GQS$_{MPC+AQ}$, against the baseline are statistically significant at level $\alpha = 0.05$ using the t-test. For the diversity results, our GQS models report notable improvements. As an example, let us take the diversity metric α-nDCG@10 to analyze the models' performance. Clearly, GQS$_{MSR+LQ}$ performs best, with a 5.55% improvement over the baseline on the AOL log. Competitive results are generated by GQS$_{MPC+LQ}$ with a 0.7% drop against GQS$_{MSR+LQ}$ but still a 4.80% improvement over the baseline. Importantly, the improvements achieved by GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ over the baseline are significant at $\alpha = 0.01$. In contrast, more modest improvements over the baseline in terms of α-nDCG@10 are achieved by GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$, both of which are significant at $\alpha = 0.05$.

In contrast, on the MSN log, we can see from Table 6.6 that the best results are again generated by GQS$_{MSR+LQ}$ but with a more modest MRR improvement against the baseline compared to that on the AOL log. In terms of MRR, significant improvements are achieved only by the GQS$_{MSR+LQ}$ model over the baseline at $\alpha = 0.05$. This can be explained by the fact that most QAC rankings generated both by the baseline and by our models are high performing, leaving limited space for significant MRR improvements of our models over the baseline. Regarding the diversity results on the MSN log, generally, our GQS models beat the baseline in terms of four metrics in Table 6.6, column 3–6. Significant improvements over the baseline in terms of these four diversity metrics are achieved at $\alpha = 0.05$ for most cases. In particular, using α-nDCG@10, GQS$_{MSR+LQ}$ achieves the largest improvement, around 3.15% over the baseline. Compared to the MRR improvements achieved by our GQS models, the improvements in terms of α-nDCG@10 are more pronounced. For some cases, redundant queries can indeed be removed from the QAC list by GQS models, resulting in improved diversity scores; however, these redundant candidates could be ranked lower than the final submitted query in the original QAC list and consequently do not affect the reciprocal rank score, and hence this has a limited impact on the MRR scores.

Summing up, based on the results achieved on the AOL and MSN logs, we conclude that our greedy query selection approach can indeed remove redundant queries in the original QAC list, returning the final submitted query early and making the final returned list cover more aspects of queries. Later, we will compare our GQS models in detail in §6.3.2 and §6.3.3.

6.3.2 Effect of the First Completion Selected by GQS

Next, we move to research question **RQ13**, for which we test our proposed GQS models on both the AOL and MSN log under different choices of the search context used, i.e., *all* preceding queries before the last query in the session or *only the last* preceding query. We begin by analyzing the results generated on the AOL log and reported in Table 6.6.

First of all, we use all previous queries before the last query in session as search context, i.e., $C_S \leftarrow \{q_1, q_2, \ldots, q_{T-1}\}$. Then, we compare the results produced by GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$ in Table 6.6 to examine the effect of the first query candidate chosen in our GQS model. Clearly, GQS$_{MSR+AQ}$ outperforms GQS$_{MPC+AQ}$ in terms of MRR and the diversity metrics. So, to some extent, our GQS model, starting with the query candidate that is semantically most similar to the current search context for D-QAC tasks, outperforms that choosing the most popular candidate first into the final list R_R. In particular, as shown in Table 6.6, on the AOL dataset, GQS$_{MSR+AQ}$ achieves an MRR improvement near 1% over GQS$_{MPC+AQ}$. For some cases, GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$ start with the same candidate, i.e., the most popular candidate is also the most semantically related one. Consequently, these two models generate the same ranked lists of query completions. As to QAC diversification, GQS$_{MSR+AQ}$ achieves very high α-nDCG@10 scores of over 0.6 and it still achieves near 2% improvement against GQS$_{MPC+AQ}$, indicating that, at the aspect level, GQS$_{MSR+AQ}$ returns more queries with multiple aspects as well as pushes the potential query to be submitted higher than GQS$_{MPC+AQ}$. Similar findings can be obtained by setting the last preceding query in a session as the search context, i.e., $C_S \leftarrow q_{T-1}$, by comparing GQS$_{MPC+LQ}$ vs. GQS$_{MSR+LQ}$. As shown in Table 6.6, compared with the difference in MRR difference between GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$, the difference in MRR between GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ is smaller. The same phenomena can be found for the α-nDCG@10 scores. However, both GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ show better performance than GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$, which motivates us to consider research question **RQ14** in §6.3.3. Thus, so far on the AOL log, we can conclude that the first query selected in our GQS model impacts the QAC ranking performance in the D-QAC tasks and our GQS model can achieve better D-QAC performance when starting with the most semantically similar query rather than the most popular.

The results achieved on the AOL log discussed in Table 6.6 are produced by averaging the scores of all prefixes at different lengths of prefixes, ranging from 1 to 5. Next, we compare our models at specific prefix lengths. For comparison, we report the results in terms of MRR and α-nDCG@10 in Table 6.7. Generally, as shown in Table 6.7, our GQS models that start with the most semantically related query, i.e., GQS$_{MSR+AQ}$ and GQS$_{MSR+LQ}$, perform better in terms of MRR than the corresponding GQS models that start with the most popular query, i.e., GQS$_{MPC+AQ}$ and GQS$_{MPC+LQ}$. Interestingly, significant improvements of the GQS models over the baseline are more easily observed at long prefixes, e.g., $\#p = 4$ and 5, than short ones, e.g., $\#p = 1$ and 2. Intuitively, longer prefixes can sharply reduce the space of candidate query completions and hence can include more similar candidates in the original QAC list, which would be pushed down in the list by our approaches, resulting in significant improvements over the baseline.

However, the MRR improvements of the GQS$_{MSR+AQ}$ model over the GQS$_{MPC+AQ}$

Table 6.7: Performance, in terms of MRR and α-nDCG@10, of GQS models under various choices of the first query candidate selected and of the search context used, at a prefix length $\#p$ ranging from 1 to 5 characters on the AOL log. The best performer per row is in boldface. Statistical significance of pairwise difference (GQS model vs. Baseline) is determined using the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for $\alpha = .01$, or $^{\triangle}/^{\triangledown}$ for $\alpha = .05$).

Metric	$\#p$	Baseline	GQS$_{MPC+AQ}$	GQS$_{MSR+AQ}$	GQS$_{MPC+LQ}$	GQS$_{MSR+LQ}$
MRR	1	.4673	.4716	.4738	.4739	**.4745**
	2	.4861	.4927	.4946	.4954	**.4960**$^{\triangle}$
	3	.5140	.5221	.5253$^{\triangle}$	.5258$^{\triangle}$	**.5263**$^{\triangle}$
	4	.5556	.5620	.5681$^{\triangle}$	.5686$^{\triangle}$	**.5689**$^{\triangle}$
	5	.5889	.5975	.6030$^{\triangle}$	.6039$^{\triangle}$	**.6045**$^{\triangle}$
α-nDCG@10	1	.6012	.6117	.6235$^{\triangle}$	.6272$^{\blacktriangle}$	**.6315**$^{\blacktriangle}$
	2	.6270	.6393	.6496$^{\triangle}$	.6551$^{\blacktriangle}$	**.6592**$^{\blacktriangle}$
	3	.6357	.6490$^{\triangle}$	.6605$^{\triangle}$	.6658$^{\blacktriangle}$	**.6713**$^{\blacktriangle}$
	4	.6611	.6758$^{\triangle}$	.6883$^{\blacktriangle}$	.6944$^{\blacktriangle}$	**.6984**$^{\blacktriangle}$
	5	.6882	.7051$^{\triangle}$	.7171$^{\blacktriangle}$	.7220$^{\blacktriangle}$	**.7276**$^{\blacktriangle}$

model and of the GQS$_{MSR+LQ}$ model over GQS$_{MPC+LQ}$ are not significant. In terms of α-nDCG@10, as reported in Table 6.7, one particular finding highlighting the difference is that, for some cases, our GQS model achieves significant improvements over the baseline in terms of α-nDCG@10 at level $\alpha = .01$, which is not the case for MRR as reported in Table 6.7. In addition, a near 2% improvement of GQS$_{MSR+AQ}$ over GQS$_{MPC+AQ}$ and around 1% improvement of GQS$_{MSR+LQ}$ over GQS$_{MPC+LQ}$ are observed in terms of α-nDCG@10. In contrast, the analogous MRR improvements in Table 6.7 are lower; this means that our models can help diversify queries.

Next, we turn our attention to results obtained using the MSN log, as reported in Table 6.6. Some findings consistent with those achieved on the AOL log can be observed: (1) the first query to be selected by the GQS approaches has a slight impact on the D-QAC performance; (2) compared to starting with the most popular query, selecting the semantically most closely related query first in the GQS models is more effective. We report on the performance at the prefix level in terms of MRR and α-nDCG@10 in Table 6.8. As shown in Table 6.8, few MRR improvements over the baseline achieved by our models at various prefix lengths is significant. However, our models do produce more diverse QAC rankings as they receive higher α-nDCG@10 scores compared to the baseline, especially when using the last query in the session as the search context. We can find from Table 6.8 that whatever the search context is used, again, selecting the most semantically related query by GQS models is more effective than injecting the most popular query into the QAC ranking list first. In addition, long prefixes (e.g., $\#p = 4$ and 5) seem to gain more in terms of diversity than short ones (e.g., $\#p = 1$ and 2), which is also confirmed by the significance tests, e.g., GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ achieve significant improvements at $\alpha = .05$ in terms of α-nDCG@10 over the baseline at $\#p = 4$ and 5 but not at $\#p = 1$. These results are consistent with those on the AOL log. Hence, apart from the conclusions established based on the AOL log, we come to another conclusion: our models perform better in cases where users continue to type more in the search box:

Table 6.8: Performance, in terms of MRR and α-nDCG@10, of GQS models under different choices of the first query candidate selected and of the search context used, at a prefix length $\#p$ ranging from 1 to 5 characters on the MSN log. The best performer per row is boldfaced. Statistical significance of pairwise difference (GQS model vs. Baseline) is determined using the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for $\alpha = .01$, or $^{\triangle}/^{\triangledown}$ for $\alpha = .05$).

Metric	$\#p$	Baseline	GQS$_{MPC+AQ}$	GQS$_{MSR+AQ}$	GQS$_{MPC+LQ}$	GQS$_{MSR+LQ}$
MRR	1	.4881	.4963	.4991$^{\triangle}$	.4995$^{\triangle}$	**.5014**$^{\triangle}$
	2	.5456	.5538	.5572$^{\triangle}$	.5578$^{\triangle}$	**.5605**$^{\triangle}$
	3	.6041	.6147	.6134	.6154	**.6152**
	4	.6523	.6616	.6647	.6646	**.6673**$^{\triangle}$
	5	.6846	.6945	.6948	.6960	**.6975**
α-nDCG@10	1	.6313	.6395	.6427	.6423	**.6431**
	2	.6564	.6637	.6691	.6708$^{\triangle}$	**.6721**$^{\triangle}$
	3	.6679	.6839$^{\triangle}$	.6893$^{\triangle}$	.6927$^{\triangle}$	**.6942**$^{\blacktriangle}$
	4	.6916	.7051	.7113$^{\triangle}$	.7132$^{\triangle}$	**.7146**$^{\triangle}$
	5	.7118	.7245	.7302$^{\triangle}$	.7332$^{\triangle}$	**.7351**$^{\triangle}$

bigger diversity gains over the baseline are achieved with long prefixes rather than with short inputs.

6.3.3 Effect of the Search Context Used by GQS

In this section, we address research question **RQ14** by changing the search context, i.e., using either the most recent query q_{T-1} as C_S in (6.3) or all preceding queries $C_S \leftarrow \{q_1, q_2, \ldots, q_{T-1}\}$. As shown in Figure 6.2a, nearly half of the sessions consist of more than two queries. In addition, we argue, in a long session with multiple queries, the query aspects of later queries may be changed as the searcher has read some results returned for previous queries and hence they may be different from the original one. The last preceding query in the search context could be a good signal of the user's updated query aspects. We first compare the results reported in Table 6.6 on the AOL and MSN logs and then move to a prefix level analysis.

We first compare the overall results of GQS$_{MPC+LQ}$ against GQS$_{MPC+AQ}$ generated on the AOL log and reported in Table 6.6. The improvements of GQS$_{MPC+LQ}$ over GQS$_{MPC+AQ}$ in terms of diversity, e.g., α-nDCG@10, are obvious but in terms of MRR they are not. For instance, GQS$_{MPC+LQ}$ shows an improvement of nearly 3% against GQS$_{MPC+AQ}$ in terms of α-nDCG@10 but less than 1% in terms of MRR. In addition, a statistically significant improvement ($\alpha = .05$) is observed in terms of α-nDCG@10 but not in terms of MRR. Thus, using only the last query as search context in our GQS models can generate notably diverse QAC ranking list on AOL. Similar findings are obtained by comparing GQS$_{MSR+LQ}$ against GQS$_{MSR+AQ}$ although the improvements are smaller. We compare the results for different search contexts at various prefix lengths in Table 6.7 in terms of MRR and α-nDCG@10, respectively. The MRR improvements of GQS$_{MPC+LQ}$ over GQS$_{MPC+AQ}$ and of GQS$_{MSR+LQ}$ over GQS$_{MSR+AQ}$ are limited but stable at different prefix lengths. However, the α-nDCG@10 results

Table 6.9: Per prefix bake-off on the AOL log, in terms of MRR and α-nDCG@10: GQS$_{MPC+LQ}$ vs. other models. The ratios (%) of test prefixes at various lengths for which GQS$_{MPC+LQ}$ loses against the corresponding model listed in row 2 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which GQS$_{MPC+LQ}$ wins have a green background.

	MRR						α-nDCG@10					
$\#p$	Baseline			GQS$_{MPC+AQ}$			Baseline			GQS$_{MPC+AQ}$		
1	26.83	37.91	35.26	17.58	60.39	22.03	22.37	24.68	52.95	15.14	58.02	26.84
2	20.72	49.17	30.11	18.06	62.15	19.79	20.36	26.83	52.81	14.65	58.97	26.38
3	14.68	60.47	24.85	16.23	63.06	20.71	18.94	27.02	54.04	12.21	60.62	27.17
4	11.38	61.24	27.38	16.08	62.97	20.95	18.51	27.63	53.86	11.45	62.78	25.77
5	10.62	62.54	26.84	15.41	61.53	23.06	18.43	28.37	53.20	09.84	64.15	26.01

Table 6.10: Per prefix bake-off on the AOL log, in terms of MRR and α-nDCG@10: GQS$_{MSR+LQ}$ vs. other models. The ratios (%) of test prefixes at various lengths for which GQS$_{MSR+LQ}$ loses against the corresponding model listed in row 2 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which GQS$_{MSR+LQ}$ wins have a green background.

	MRR						α-nDCG@10					
$\#p$	Baseline			GQS$_{MSR+AQ}$			Baseline			GQS$_{MSR+AQ}$		
1	24.37	36.53	39.10	17.91	63.31	18.78	20.12	21.83	58.05	17.60	62.61	19.79
2	19.65	47.86	32.49	16.47	64.39	19.14	20.21	23.07	56.72	17.08	63.75	19.17
3	13.78	57.85	28.37	15.93	66.87	17.20	19.55	24.38	56.07	16.35	65.05	18.60
4	12.84	59.17	27.99	15.55	67.62	16.83	18.42	25.82	55.76	15.21	66.36	18.43
5	11.53	61.02	27.45	14.41	68.15	17.44	18.59	27.03	54.38	15.57	67.32	17.11

set GQS$_{MPC+LQ}$ apart from GQS$_{MPC+AQ}$ with significant improvements ($\alpha = .05$) except for $\#p = 5$. In contrast, GQS$_{MSR+LQ}$ and GQS$_{MSR+AQ}$ yield very similar α-nDCG@10 scores. We attribute these findings to the fact that: (1) diverse queries can be returned by our GQS models usually at positions lower than the final submitted query, resulting in indistinguishable MRR scores but modified diversity scores; (2) half of the sessions consist of only two queries, see Figure 6.2a, which means that the search contexts used are the same, resulting in many ties. To verify this claim, we compare GQS$_{MPC+LQ}$ vs. GQS$_{MPC+AQ}$ and GQS$_{MSR+LQ}$ vs. GQS$_{MSR+AQ}$ as well as the baseline in a per prefix bake-off. We report the results in Table 6.9 and 6.10, respectively.

From Table 6.9, we see that, against the baseline, GQS$_{MPC+LQ}$ wins many comparisons, especially in terms of α-nDCG@10 ($> 50\%$). We also find many ties between the baseline and GQS$_{MPC+LQ}$ in terms of MRR. In contrast, GQS$_{MPC+LQ}$ yields a majority of draws against GQS$_{MPC+AQ}$ in terms of both MRR and α-nDCG@10. Some of the draws occur when GQS$_{MPC+LQ}$ and GQS$_{MPC+AQ}$ return the same ranked list of query completions; others happen on prefixes for which the two models return the final submitted query at top positions in the list of query completions, e.g., 1 or 2. As to a comparison of GQS$_{MSR+LQ}$ vs. GQS$_{MSR+AQ}$, similar results can be found except that there are more ties in terms of MRR and α-nDCG@10. One particularly interest-

Table 6.11: Per prefix bake-off on the MSN log, in terms of MRR and α-nDCG@10: GQS$_{MPC+LQ}$ vs. other models. The ratios (%) of test prefixes at various lengths for which GQS$_{MPC+LQ}$ loses against the corresponding model listed in row 2 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which GQS$_{MPC+LQ}$ wins have a green background.

	MRR						α-nDCG@10					
$\#p$	Baseline			GQS$_{MPC+AQ}$			Baseline			GQS$_{MPC+AQ}$		
1	25.14	39.43	35.43	16.17	63.51	20.32	27.65	21.75	50.60	14.07	62.34	23.59
2	18.62	50.83	30.55	15.57	65.01	19.42	25.18	22.82	52.00	13.26	64.13	22.61
3	13.69	62.45	23.86	13.69	67.87	18.44	26.74	23.95	49.31	11.33	66.22	22.45
4	12.79	63.53	23.68	13.16	68.69	18.15	22.14	26.07	51.79	10.76	67.66	21.58
5	13.02	64.68	22.30	12.88	69.79	17.33	25.28	26.73	47.99	10.01	67.79	22.20

Table 6.12: Per prefix bake-off on the MSN log, in terms of MRR and α-nDCG@10: GQS$_{MSR+LQ}$ vs. other models. The ratios (%) of test prefixes at various lengths for which GQS$_{MSR+LQ}$ loses against the corresponding model listed in row 2 have a red background, ratios with equal performance have a yellow background, and those of prefixes for which GQS$_{MSR+LQ}$ wins have a green background.

	MRR						α-nDCG@10					
$\#p$	Baseline			GQS$_{MSR+AQ}$			Baseline			GQS$_{MSR+AQ}$		
1	22.63	40.77	36.60	14.33	64.32	21.35	28.39	23.43	48.18	15.37	63.07	21.56
2	15.32	51.75	32.93	13.57	65.33	21.10	23.73	23.24	53.03	15.10	64.12	20.78
3	12.27	62.68	25.05	12.23	68.11	19.66	23.85	25.07	51.08	14.22	65.91	19.87
4	11.48	63.93	24.59	12.54	68.79	18.67	23.77	25.87	50.36	14.51	66.74	18.75
5	10.65	65.11	24.24	11.53	69.70	18.77	22.26	26.31	51.43	13.24	67.35	19.41

ing point shown in Table 6.9 and 6.10 is that, for most cases, more ties occur when the prefix becomes longer. This is because the QAC models can return the final submitted query early on for long prefixes and hence they generate more similar rankings of query completions.

The outcomes of the main comparisons on the MSN log, i.e., GQS$_{MPC+LQ}$ vs. GQS$_{MPC+AQ}$ and GQS$_{MSR+LQ}$ vs. GQS$_{MSR+AQ}$ are consistent with those on the AOL log. Regarding the prefix level analysis, see Table 6.8, although the GQS models using the last query as search context still beat the corresponding models that use all preceding queries in terms of MRR and α-nDCG@10, the improvements are not statistically significant. Some significant improvements of GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ against the baseline are observed, especially on long prefixes. Similarly, we report on a per prefix bake-off in terms of MRR and α-nDCG@10 in Table 6.11 and 6.12. As the MSN log contains more sessions with only two queries than the AOL log, see Figure 6.2a, more draws are found between GQS$_{MPC+LQ}$ vs. GQS$_{MPC+AQ}$ as well as GQS$_{MSR+LQ}$ vs. GQS$_{MSR+AQ}$. This simply happens because for two-query sessions, the search context used in GQS models consisting either of all preceding queries or only of the last query are always the same.

Table 6.13: Agreements (%) between side-by-side comparisons by humans and per prefix bake-offs by algorithms.

	GQS$_{MPC+LQ}$ vs.		GQS$_{MSR+LQ}$ vs.	
$\#p$	Baseline	GQS$_{MPC+AQ}$	Baseline	GQS$_{MSR+AQ}$
1	89.20	85.20	92.60	83.40
2	90.20	86.20	93.40	84.60
3	91.40	87.40	94.40	85.40
4	92.40	88.40	95.00	87.00
5	94.00	90.60	96.00	88.20

6.3.4 Side-by-side Experiments

To answer **RQ15**, we follow the set-up in (Chapelle et al., 2012) and investigate the agreement between the side-by-side comparison produced by human judges and the relative ranking of QAC approaches that results from the bake-offs discussed in §6.3.3. See Table 6.13. We find that the two evaluation methodologies point in the same direction in the vast majority of pairwise comparisons: the agreement ranges between 83% and 96%. For instance, for the comparison between GQS$_{MPC+LQ}$ and the baseline, we find that human preferences agree with the preferences obtained from the bake-offs in §6.3.3 in more than 90% of the cases. The agreement between the two types of preference for GQS$_{MPC+LQ}$ and GQS$_{MPC+AQ}$ are somewhat lower: the difference in performance in terms of diversity between GQS$_{MPC+LQ}$ and GQS$_{MPC+AQ}$ is smaller than between GQS$_{MPC+LQ}$ and the baseline, making it harder for human judges to identify differences or to identify the direction of the difference. We also observe that agreement tends to be higher for longer prefixes: with longer prefixes, the number of possible completions is smaller than for shorter prefixes, reducing the possibilities for disagreement and making it easier for both systems and humans to determine which aspects and completions are relevant.

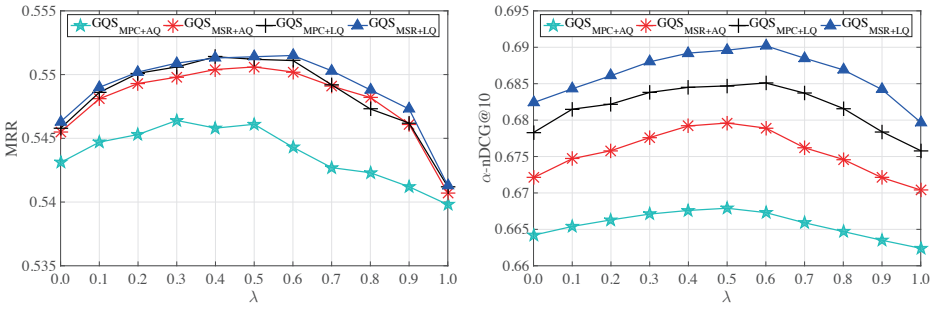
Given the high levels of agreement between human preferences and preferences induced from contrastive experiments, we conclude that the (significant) differences between QAC approaches that we found in §6.3.3 are confirmed by the side-by-side experiments.

6.3.5 Impact of Parameter Tuning

In this section, we conduct a parameter sensitivity analysis of our GQS models. We examine the performance of our GQS models in §6.3.5 by changing the trade-off parameter λ in (6.4) and by varying the number of latent features k_f used for Bayesian probabilistic matrix factorization in §6.3.5, and then see how the models perform when more (or fewer) query completions are returned by varying the cutoff N in §6.3.5.

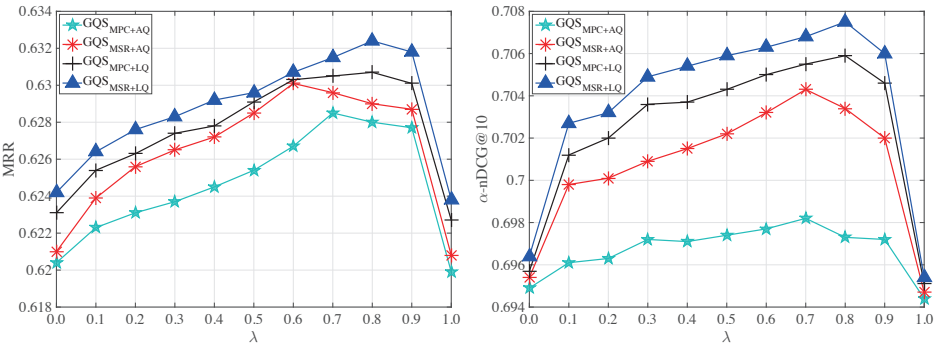
Zooming in on the trade-off parameter λ in (6.4)

We first examine the overall performance of our GQS models in terms of MRR and α-nDCG@10 by gradually changing the trade-off parameter λ from 0 to 1 with steps of



(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.3: Effect on D-QAC performance of GQS models in terms of MRR (left) and α-nDCG@10 (right) by changing the trade-off λ in (6.4), tested on the AOL log.



(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.4: Effect on D-QAC performance of GQS models in terms of MRR (left) and α-nDCG@10 (right) by changing the trade-off λ in (6.4), tested on the MSN log.

0.1, and then plot the results in Figure 6.3 and Figure 6.4 for the AOL and MSN logs, respectively.

On the AOL log, we can see, from Figure 6.3a, that when λ varies from 0 to 0.3, the MRR scores of all GQS models increase; they continue to go up until $\lambda = 0.5$ except for one case where GQS_{MPC+AQ} displays an MRR decrease after $\lambda = 0.3$. For most GQS models the performance in terms of MRR goes down when λ changes from 0.5 to 1. For any GQS model, if it only focuses on search popularity, i.e., $\lambda = 1$ in (6.4), the performance is worse than when it only focuses on the search context, i.e., $\lambda = 0$ in (6.4). In terms of α-nDCG@10, the peak performance appears near $\lambda = 0.5$ for GQS_{MPC+AQ} and GQS_{MSR+AQ} or near $\lambda = 0.6$ for GQS_{MPC+LQ} and GQS_{MSR+LQ}. For any λ, GQS_{MSR+LQ} always performs best among the four models in terms of both MRR and α-nDCG@10.

In contrast, for MRR on the MSN log, GQS_{MPC+LQ} and GQS_{MSR+LQ} favor a large

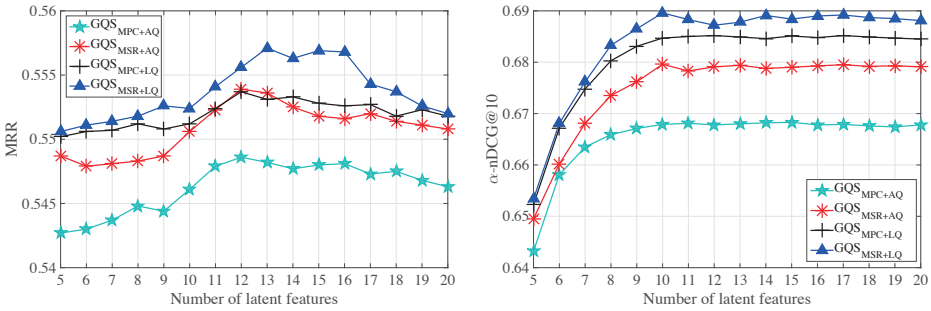
λ. For instance, they achieve peak MRR scores near $\lambda = 0.8$. However, GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$ prefer a somewhat smaller λ than GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$. As shown in Figure 6.4a, a maximal MRR score is returned for GQS$_{MSR+AQ}$ near $\lambda = 0.6$ and near $\lambda = 0.7$ for GQS$_{MPC+AQ}$. For the performance in terms of α-nDCG@10, a sharp increase is observed when λ changes from 0 to 0.1 on all four models as shown in Figure 6.4b. This means that the search context does help to diversify the query candidates. In addition, the α-nDCG@10 scores go up until $\lambda = 0.8$ for GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$ and $\lambda = 0.7$ for GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$. All four GQS models present a relatively low score when $\lambda = 1.0$. In addition, compared to GQS$_{MPC+AQ}$ and GQS$_{MSR+AQ}$, the other two GQS models show bigger fluctuations in terms of both MRR and α-nDCG@10 when λ changes.

From the observations in Figure 6.3 and Figure 6.4, we can conclude that: (1) in our GQS models for the diversified query auto completion task, search popularity and search context are both important for query diversification. Compared to search popularity, search context may contribute much more to the effectiveness of GQS models for diversified query auto completion as a larger λ ($0.5 < \lambda < 0.9$) results in better performance than that of $0.1 < \lambda < 0.4$, especially on the MSN log, see Figure 6.4; (2) the search context used in GQS models, i.e., either only the last query or all preceding queries in session, has a small impact on the performance as these four GQS models show their peak performance at various λ; (3) λ may exert a bigger influence on α-nDCG@10 than on MRR as relatively noticeable margins can be seen when λ changes shown in Figure 6.3b and Figure 6.4b.

Effect of the number of latent features k_f uses in Bayesian probabilistic matrix factorization

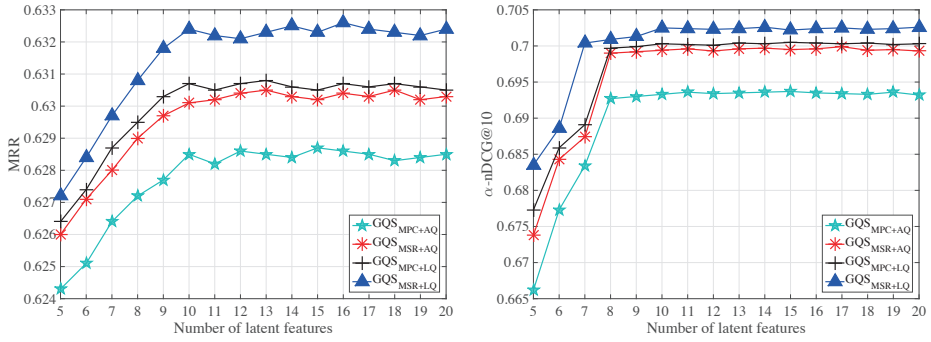
Next, we zoom in on the number of latent features k_f used in Bayesian probabilistic matrix factorization for generating the query distributions over aspects. We manually vary the value of k_f in GQS models from 5 to 20. See Figure 6.5 on the AOL log and Figure 6.6 on the MSN log, respectively.

Generally, for the AOL log, when the number of latent features k_f used in BPMF increases from 5 to 12, the performance of our GQS models increases dramatically in terms of MRR, with a little fluctuation. However, the α-nDCG@10 scores stop increasing for $k_f \geq 10$. In addition, when the number of latent features k_f varies from 10 to 20, the performance of our GQS models seems to level off, especially in terms of α-nDCG@10. Another important finding is that the performance in terms of MRR sometimes goes down when k_f increases. For instance, when k_f varies from 18 to 20, the MRR scores of the GQS models except GQS$_{MPC+LQ}$ drops. For the MSN log, the MRR scores of the GQS models invariably increase from $k_f = 5$ to 10 for all GQS models and remain stable for $k_f = 10, \ldots, 20$. Compared to the MRR results on the AOL log (in Figure 6.5a), the GQS models seem to be more sensitive to the number of latent features on the MSN log. When k_f is small, e.g., $5 < k_f < 10$, the MRR jumps (see Figure 6.6a) are easily observed, especially for GQS$_{MSR+LQ}$ and GQS$_{MPC+LQ}$. Regarding α-nDCG@10 on the MSN log, similar findings can be observed except that all GQS models arrive at a stable level of performance much earlier (when $k_f = 8$) than on the AOL log (when $k_f = 10$). From the results shown in Figure 6.5 and Figure 6.6, we conclude that our GQS models



(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.5: Effect on diversified query auto completion performance of GQS models in terms of MRR (left) and α-nDCG@10 (right), tested on the AOL log, by changing the number of latent features used in BPMF.



(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.6: Effect on diversified query auto completion performance of GQS models in terms of MRR (left) and α-nDCG@10 (right), tested on the MSN log, by changing the number of latent features used in BPMF.

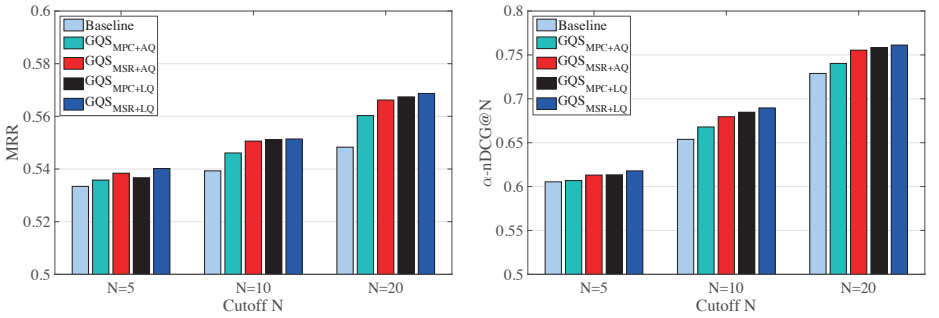
are robust and not sensitive to the number of latent features k_f when it is "large enough", e.g., $k_f > 10$.

Zooming in on the cutoff N

Finally, we examine the performance of our GQS models and the baseline, i.e., QDQCR, when less (or more) query completions are finally returned by setting the cutoff $N = 5$ (or $N = 20$). We plot the results in terms of MRR and α-nDCG@N scores ($N = 5, 10, 20$) in Figure 6.7 and Figure 6.8, tested on the AOL log and the MSN log, respectively.[8]

As shown in Figure 6.7 and Figure 6.8, for all five models on both logs, the overall performance in terms of MRR increases when more query completions are initially re-

[8]The results for $N = 10$ were already partially reported in Table 6.6.



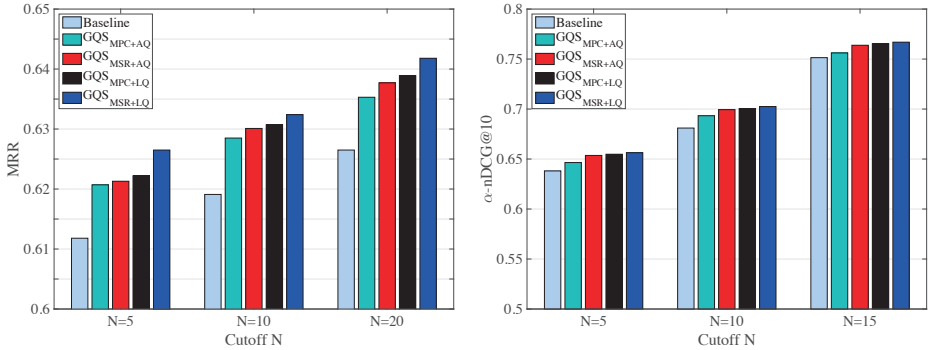
(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.7: D-QAC performance of all discussed models, tested on the AOL log, in terms of MRR (left) and α-nDCG@10 (right) when more (or less) query completions are returned. Note: the scales are different.

turned for re-ranking, i.e., when N becomes larger. A larger value of N simply increases the probability of including the ground truth in the QAC list. In particular, on both the AOL and MSN log, these models report competitive MRR score when $N = 5$. Moreover, the MRR improvements realized by our GQS models over the baseline are further magnified as N goes up. For instance, on the AOL log, GQS$_{MSR+LQ}$ results in a 1.27% MRR improvement over the baseline at $N = 5$, a 2.24% improvement at $N = 10$, and a 3.72% improvement at $N = 20$. With respect to query diversification, the improvements of the GQS models are more obvious in terms of α-nDCG@N ($N = 5$, 10, and 20) than MRR as indicated by the relative improvements over the baseline. For instance, at cutoff $N = 20$, GQS$_{MSR+LQ}$ shows a 4.43% improvement over the baseline in terms of α-nDCG@20. This may be because not too many redundant queries can be found among the top 5 candidates in the list of query completions, hence it is difficult to make significant improvements over the baseline at cutoff $N = 5$. However, as more candidates are returned, more query redundancy is introduced into the list of query completions, and it becomes easier for the GQS models to improve over the baseline. Therefore, based on our findings from Figure 6.7 and Figure 6.8, we conclude that compared to the baseline, the advantages of our GQS models over the baseline become more prominent when more query completions are returned.

6.4 Conclusion

In this chapter, we have proposed the challenge of diversifying query auto completion. We believe this can help search engine designers especially in settings with a limited number of query completions. To address the query auto completion diversification task, we propose a greedy query selection (GQS) model and use the ODP taxonomy to identify aspects of URLs, based on which we then assign query aspects via clickthrough data derived from query logs. The problems of data sparsity and cold-start in traditional recommendation systems are overcome by incorporating a Bayesian probabilistic matrix



(a) Performance in terms of MRR. **(b) Performance in terms of α-nDCG@10.**

Figure 6.8: D-QAC performance of all discussed models, tested on the MSN log, in terms of MRR (left) and α-nDCG@10 (right) when more (or less) query completions are returned. Note: the scales are different.

factorization approach and determining the semantically most closely related query using word2vec, respectively.

We have experimentally investigated the diversified query auto completion performance of our GQS models under various settings. Our results show that the GQS model performs best when starting with the semantically most closely related query completion and using only the last preceding query in a session as the search context. This finding indicates that query aspects are commonly shared by successive queries in a session and may be changed especially in long sessions with multiple queries. In addition, we have found that our GQS model performs better when more query completions are fed to it.

As future work, we plan to move our models to other datasets, where the ground truth of query aspects is to be generated by humans rather than the automatically generated clickthrough data used in this paper. In addition, it would be interesting to introduce other scenarios to deal with the query cold-start problem as this may be helpful to obtain the query distribution over aspects. Furthermore, since we only consider query aspects as they are expressed, either explicitly or implicitly, by current search popularity or previously submitted queries, it would be interesting to further collect users' long term search history so as to enhance the performance of diversifying query auto completion, thereby personalizing diversified query auto completion, which can help narrow the space of query topics, i.e., adjust the amount of diversification, by removing non-relevant queries, but can still promote relevant queries as well as diversify QAC completions.

In Chapter 4, we have focused on user's short- and long-term search context for personalized query auto completion. In this chapter, we have explored information from the search context in the current session for diversifying query auto completion. In the next chapter (Chapter 7), we will investigate whether personalization can always improve the performance of query auto completion.

7

Selectively Personalizing Query Auto Completion

To cater for a user's specific information needs, personalized query auto completion (QAC) strategies have been investigated that take either the user's search history or their user profile into account. For instance, as studied in Chapter 4, the user's long- and short-term search context can be used for personalized query auto completion; and in Chapter 6 the search history in current session is explored to diversifying query auto completion. Such methods personalize the list of query completions in the same manner. However, it is unclear whether personalization is consistently effective under different search contexts. In our final research chapter, we study the QAC problem by selectively personalizing the list of query completions. Based on a lenient personalized QAC strategy that basically encodes the ranking signal as a trade-off between query popularity and search context, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model to study such a trade-off. In particular, we predict an effective trade-off in each case based on a regression model, where the typed prefix, the clicked documents and the preceding queries in session are considered for weighing personalization in QAC.

It has been shown that, in general web search (Dou et al., 2007; Teevan et al., 2008) and recommendation systems (Zhang et al., 2013), not all queries should be personalized equally as personalization strategies occasionally harm the search accuracy. Similarly, in a QAC task, we propose that prefixes should not be handled in the same personalization manner because: (1) user's initial information needs may be addressed by previous interactions; (2) users may change their search intent during a session. Such clues can be explicitly expressed by the clicks or directly revealed by the query flow in a session.

In this chapter, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model to re-rank the top N query completions produced by the MPC (Most Popular Completion) model (Bar-Yossef and Kraus, 2011). In particular, personalization in the proposed SP-QAC model is individually weighted when being combined with ranking signals from search popularity. We study the following factors for weighing personalization: the typed prefix for which we recommend query suggestions, the clicked documents for inferring user's satisfaction and the topic changes of preceding queries in session for detecting search intent shifts. We use the description of each URL from the ODP (Open Directory Project) data[1] to represent documents and queries based on the

[1] `http://www.dmoz.org`

word2vec model (Mikolov et al., 2013b) when inferring a user's satisfaction as well as query intent shifts in session. In doing so, we answer the following research questions:

RQ17 Does selective personalization scheme help improve the accuracy of ranking query completions in a generic personalized QAC approach?

RQ18 In the SP-QAC model, what is the impact on its performance of varying the inputs to the regression model?

Finally, we quantify the improvement in terms of Mean Reciprocal Rank (MRR) of our proposal over state-of-the-art QAC baselines on a publicly available query log dataset. We find the SP-QAC model, which selectively outweighs or depresses the contribution of personalization in a generic QAC approach, outperforms a traditional non-personalization QAC approach and a uniformly personalized QAC approach with a fixed trade-off controlling the contribution of search popularity and search context.

Our contributions in this chapter are summarized as:

1. We propose a Selectively Personalizing Query Auto Completion (SP-QAC) model that flexibly outweighs or depresses the contribution of personalization in QAC.
2. We study the typed prefix, the clicked documents and the preceding queries in session to estimate the weight of personalization in a QAC task.

The remainder of this chapter is organized as follows. We detail our approach for selectively personalizing query auto completion in §7.1. Our experimental setup is presented in §7.2. In §7.3, we provide the results as well as the discussions. We conclude in §7.4.

7.1 Approach

As discussed in Chapters 2, 4, 5 and 6, a straightforward approach to ranking query completions is based on the popularity of queries. Bar-Yossef and Kraus (2011) refer to this type of ranking as the Most Popular Completion (MPC) model:

$$MPC(p) = \underset{q \in S(p)}{\operatorname{argmax}}\, w(q),\ w(q) = \frac{f(q)}{\sum_{i \in L} f(i)}, \tag{7.1}$$

where $f(q)$ denotes the number of occurrences of query q in search log L, and $S(p)$ is a set of query completions that start with prefix p.

To cater for a user's particular information need, personalization is incorporated into the MPC model. As described in (Bar-Yossef and Kraus, 2011; Cai et al., 2014b), a generic personalized QAC approach basically employs a fixed parameter λ to control the contribution of personal information to generate the final ranking of query completions. For instance, Bar-Yossef and Kraus (2011) compute a hybrid score for each query candidate q_c, which is a convex combination of two scores, i.e., a query popularity score $MPCsco(q_c)$ and a personalization score $Psco(q_c)$:

$$hybsco(q_c) = \lambda \cdot MPCsco(q_c) + (1 - \lambda) \cdot Psco(q_c), \tag{7.2}$$

where $MPCsco(q_c)$ is estimated by candidate q_c's frequency in the query log and $Psco(q_c)$ is measured by q_c's similarity to the search context in session. Such scenarios handle each prefix uniformly. However, users may modify their search intent in a session. As a consequence, personalization may harm the accuracy of QAC ranking if we continue to

use the previous search context. Hence, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model, which emphasizes on personalization inconsistently in different cases when generating the final hybrid score of a candidate as follows:

$$hybsco(q_c) = \phi(\cdot) \cdot MPCsco(q_c) + (1 - \phi(\cdot)) \cdot Psco(q_c), \tag{7.3}$$

where the function $\phi(\cdot)$ outputs a trade-off in [0,1] and is parameterized by the typed prefix, the clicked documents and the preceding query in session, which are to be described in following sections. By doing so, personalization in QAC can be individually weighted in each particular case.

7.1.1 Signal from the Typed Prefix

As described in Chapter 2, most previous work on query auto completion (Bar-Yossef and Kraus, 2011; Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013) only considers the typed prefix for generating a list of matched query candidates, ignoring the potential signal hidden in the typed prefix for personalization in QAC. However, the typed prefix normally reveals a strong clue for inferring user's personal query activity, such as query expansion and query repetition, etc. Thus, it can be introduced for weighing personalization in a QAC task.

Intuitively, if a user's input to the search box appears as a prefix of query terms in previous queries inside the current search session, it is possible that this user repeats their query (Jiang et al., 2014b). As a consequence, personalization can make sense. Thus, we capture the signal from the typed prefix p for personalization by introducing a factor f_p to weighing the personalization for query auto completion as follows:

$$f_p = \frac{|\mathcal{W}(p)|}{|\mathcal{S}|} + c, \tag{7.4}$$

where $|\mathcal{W}(p)|$ and $|\mathcal{S}|$ indicate the number of words that start with p and that appear in session, respectively; c is a small constant for smoothing. A larger value of f_p could imply a higher importance of personalization in the final ranking score in (7.3).

7.1.2 Inferring Search Satisfaction from Clicked Documents

User's clicks on retrieved documents in return to a query are a widely used behavioral signal for measuring search satisfaction (Kim et al., 2014), which can be further measured by the closeness between a submitted query and its clicked documents, because the closer they are the more satisfied the user could be (Fox et al., 2005). Cosine similarity can be applied to model a factor f_d for measuring closeness:

$$f_d = \begin{cases} c, & \text{no clicks} \\ \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} \frac{1}{|\mathcal{D}_q|} \sum_{d \in \mathcal{D}_q} \cos(q, d), & \text{otherwise,} \end{cases} \tag{7.5}$$

where $\mathcal{D}_q$ is a set of clicked documents corresponding to a submitted query q in a set $|Q|$ of previous queries in session. Each clicked document $d \in \mathcal{D}_q$ has a short description text T extracted from the ODP data. We vectorize this document description consisting of a

sequence of words using the word2vec model (Mikolov et al., 2013b) where each word is represented by a vector v_w. In doing so, a document can be vectorized by averaging the words in T as $d = \frac{1}{|T|}\sum_{v_w \in T} v_w$. After that, a query q is then similarly represented by averaging its clicked documents $\mathcal{D}$ in the training log, i.e., $q = \frac{1}{|\mathcal{D}|}\sum_{d \in \mathcal{D}} d$. In practice, for a query q that has no clicked documents, the same representation of its most semantically similar query q_o, which is identified by the word2vec model (Mikolov et al., 2013b) and has been vectorized in the training period, is assigned to it by

$$q_o \leftarrow \underset{q_l \in Q_L}{\operatorname{argmax}} \cos(q, q_l) = \underset{q_l \in Q_L}{\operatorname{argmax}} \frac{1}{W} \sum_{w_k \in q} \sum_{w_j \in q_l} \cos(w_k, w_j),$$

where Q_L is a set of queries that have clicked documents in the training period and W is the number of word pairs between two queries.

Intuitively, a large score of f_d in (7.5) indicates a high probability that user is satisfied with the results, thus resulting in a low weight of personalization in QAC. We assign a small constant c to f_d when no clicks are available, where personalization could make sense because the user's request has not been addressed and they may continue to submit similar queries.

7.1.3 Detecting Topic Shifts From Preceding Queries

Generally, a long session may contain multi-topical queries in web search, that often happens when user's search task changes in long sessions (Jiang et al., 2014a). We perceive such signals for weighing personalization in a QAC task by the topic shifts of submitted queries in a query flow $\{q_1, q_2, \ldots, q_{t-1}, q_t\}$ in session. A strong topical shift of preceding queries implies low-weight personalization in QAC because the user may be moving to other topics. Hence, we model a factor f_q introduced by the topic shifts of preceding queries in session as follows:

$$f_q = \begin{cases} c, & r = 1 \text{ or } 2 \\ \cos(q_1, q_2), & r = 3 \\ \cos(q_{r\text{-}1} - q_{r\text{-}2}, q_{r\text{-}2} - q_{r\text{-}3}), & r > 3, \end{cases} \tag{7.6}$$

where r is the query position in the session and each query is vectorized by the scheme described in §7.1.2. For queries at the beginning of a session, i.e., $r = 1$ or 2, the topic shift from queries is unavailable, making no impact on personalization, and thus we assign a small constant c to f_q. Similarly, at query position $r = 3$, the relative topical shift of queries is still unavailable. Instead, we use the absolute query similarity between $q_{r\text{-}1}$ and $q_{r\text{-}2}$ as an indicator of query topical shift. For queries at position $r > 3$, we study the topic shift from their preceding queries, i.e., $q_{r\text{-}1}$, $q_{r\text{-}2}$ and $q_{r\text{-}3}$.

Essentially, a large value of f_q is produced by high topical similarity of the preceding queries, implying a reasonably high probability that the user's search intent has persisted from previous queries to the current session. From this point of view, for a QAC task, personalization should be outweighed.

7.1.4 Weighing Personalization

Taking these discussed factors into account, i.e., the typed prefix, the clicked documents and the preceding queries, we adopt logistic regression to model how likely each factor affects the weight of personalization in a QAC task. In the training period, for each typed prefix, we manually change the value of λ in (7.3) from 0 to 1 with an increasing interval 0.1 to guarantee the final submitted query at the top position. By doing so, we can get an optimal weight for personalization, which is used as a label in the regression model.

Regarding the inputs to the regression model, the factors discussed above, i.e., f_p, f_d and f_q, are involved. To overcome the noise brought by the typed prefix, we implement the regression model on various inputs conditional on whether the terms in previous queries start with the prefix. Hence, the selective weight $\phi(\cdot)$ in (7.3) is finally generated as follows:

$$\phi(\cdot) = \begin{cases} Reg(f_d, f_q), & \text{if } f_p = c \\ Reg(f_p, f_d, f_q), & \text{otherwise.} \end{cases} \tag{7.7}$$

We use the personalization scenario proposed in (Cai et al., 2014b) to compute the $Psco(q_c)$ score in (7.3) when generating the optimal personalization weights.

7.2 Experimental Setup

In this section, we present the experimental setup and the baselines for comparisons. Following the setup described in Chapter 6, we use the publicly available AOL query log dataset in our experiments, which is split into three parts: a training set, a validation set and a test set consisting of the first 60%, the following 20% and the last 20% of the query log, respectively. One-query and no-click sessions are both excluded in our experiments as not enough search context is available. In addition, we only keep the cases where the final submitted query is included in the top N query completions returned by the MPC approach, which follows the previous QAC work and is a commonly used methodology in QAC tasks (Cai et al., 2014b; Jiang et al., 2014b; Shokouhi, 2013).

For comparison, the following baselines are selected: (1) the most popular completion (MPC) approach which ranks query candidates by their frequency, referred to as MPC, which we have also used in Chapters 4, 5 and 6; (2) a personalized QAC approach based on session context with a fixed trade-off $\lambda = 0.5$ in (7.2), denoted as P-QAC (Cai et al., 2014b). As before, Mean Reciprocal Rank (MRR) is used for evaluating the performance of QAC models. Statistical significance of observed differences between the performance of two approaches is tested using a two-tailed paired t-test and is denoted using $^{\blacktriangle}/^{\blacktriangledown}$ for significant differences for $\alpha = .01$ and $^{\triangle}/^{\triangledown}$ for $\alpha = .05$.

In addition, we set $N = 10$ in our experiments, which means that the top ten query completions returned by the MPC approach are to be re-ranked. We randomly assign a small value 0.01 to the constant c in our experiments.

7.3 Results and Discussion

In this section, we report on our experiments to verify the effectiveness of our proposed SP-QAC model. In §7.3.1, we examine the general performance of our proposal; and

Table 7.1: Performance of QAC models in terms of MRR at a prefix length $\#p$ ranging from 1 to 5 characters. The best performer per row is highlighted. Statistical significance of pairwise differences of SP-QAC vs. MPC and SP-QAC vs. P-QAC are detected by the t-test ($^{\blacktriangle}$/$^{\blacktriangledown}$ for α = .01, or $^{\triangle}$/$^{\triangledown}$ for α = .05) and marked in the upper left and upper right hand corners of SP-QAC scores, respectively.

$\#p$	MPC	P-QAC	SP-QAC
1	0.5368	0.5422	$^{\triangle}$**0.5535**$^{\triangle}$
2	0.5556	0.5628	$^{\triangle}$**0.5744**$^{\triangle}$
3	0.5944	0.6046	$^{\triangle}$**0.6165**
4	0.6294	0.6427	$^{\blacktriangle}$**0.6547**
5	0.6589	0.6646	$^{\blacktriangle}$**0.6762**

in §7.3.2, we zoom in on the particular factors for weighing personalization in our QAC model.

7.3.1 Performance of the SP-QAC Model

To answer the research question **RQ17**, we compare the results of our proposed model, i.e., the SP-QAC model, with that of the baselines. We report the results in Table 7.1.

Clearly, as shown in Table 7.1, a generic personalization scheme helps improve the QAC performance in terms of MRR as the MRR scores of the P-QAC model are higher than those of the MPC approach at each particular prefix length. In addition, when a selective personalization strategy is embedded into the P-QAC model, the QAC performance is boosted further as the MRR scores of the SP-QAC model are increased upon those of the baselines, i.e., the MPC and P-QAC models. Compared to the MPC approach, significant MRR improvements of the SP-QAC approach are observed at level $\alpha = .05$ for the short prefixes, e.g., $\#p = 1$ or 2, and at level $\alpha = .01$ for the long prefixes, e.g., $\#p = 4$ or 5. This difference can be explained by the fact that for most cases, compared to short prefixes, long prefixes reveal a stronger signal for search personalization, like query repetition. However, compared to the results of the P-QAC model, significant MRR improvements of the SP-QAC model are only observed at short prefixes, i.e., $\#p = 1$ or 2. This is due to the fact that, compared to short prefixes, long prefixes often return the correct query early in the list of query completions in both models, making it difficult to gain any further performance improvements from selective personalization.

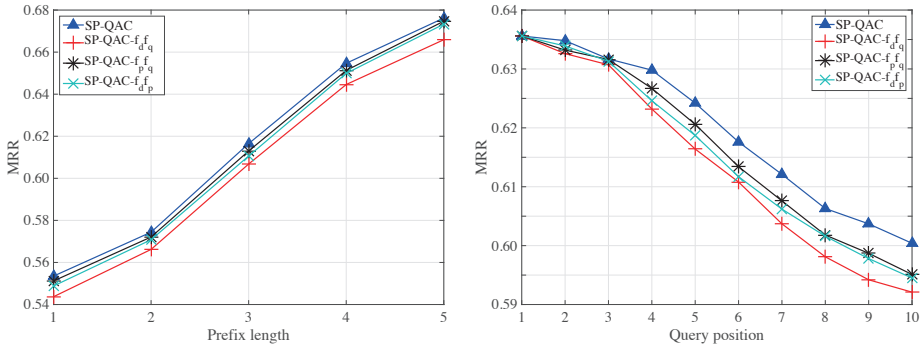
To further verify the effectiveness of the selective personalization scheme for QAC, we examine the performance of QAC models at different query positions, i.e., at the beginning (1, 2 or 3), in the middle (4, 5 or 6) and in the later (> 6) part of a session. We present the results in Table 7.2. We can see that at the start of a session, these three models return competitive MRR scores as limited information from search context is available for the P-QAC and SP-QAC models. However, as the search context becomes richer, significant improvements in terms of MRR are obtained by the SP-QAC model over the MPC and P-QAC models, respectively.

Table 7.2: Performance of QAC models in terms of MRR at various query positions. The best performer per row is highlighted. Statistical significance of pairwise differences of SP-QAC vs. MPC and SP-QAC vs. P-QAC are detected by the t-test ($^{\blacktriangle}/^{\blacktriangledown}$ for α = .01, or $^{\triangle}/^{\triangledown}$ for α = .05) and marked in the upper left and upper right hand corners of SP-QAC scores, respectively.

Query position	MPC	P-QAC	SP-QAC
$\{1, 2, 3\}$	0.6283	0.6327	**0.6340**
$\{4, 5, 6\}$	0.6005	0.6113	$^{\blacktriangle}$**0.6257**$^{\triangle}$
$\{7, 8, 9, \cdots\}$	0.5737	0.5863	$^{\blacktriangle}$**0.6058**$^{\triangle}$

7.3.2 Zoom in on Factors for Weighing Personalization

Next, we turn to the research question **RQ18** and examine the performance of the SP-QAC model under different inputs to the regression model for generating the weights of personalization in a QAC model. We manually remove one factor and remain the other two for regression model, resulting in the SP-QAC-$f_d f_q$, SP-QAC-$f_p f_q$ and SP-QAC-$f_d f_p$ models, which correspond to the SP-QAC model without considering the factors f_p, f_d and f_q for selectively personalizing QAC, respectively. We plot the results of the SP-QAC models in Figure 7.1, including the model incorporating all three factors for selective personalization which is denoted by SP-QAC.



(a) QAC performance at various prefix length.

(b) QAC performance at various query position.

Figure 7.1: Performance in terms of MRR of the SP-QAC models under different schemes for weighing personalization, tested at various prefix lengths (left) and at various query positions (right).

Generally, as shown in Figure 7.1, the SP-QAC model achieves the best performance in terms of MRR at any prefix length and any query position. In particular, as shown in Figure 7.1a, as the prefix length increases, the MRR scores increase monotonically because a long prefix can sharply cut down the space of possible completions matching the typed prefix, resulting in increased MRR scores. In contrast, as shown in Figure 7.1b, the MRR scores of the SP-QAC models are decreasing when users continue to query in

a session. This can be attributed to that, in the later part of a search session, users are inclined to submit uncommon queries without clear purposes, making it difficult to return the correct completion early by the MPC model, which our proposal partially depends on.

Next, we zoom in on the three factors considered in selective personalization for QAC. As shown in Figure 7.1a, the MRR scores of the SP-QAC-$f_p f_q$ and SP-QAC-$f_d f_p$ models approximate those of the SP-QAC model when the prefix length goes up. However, the MRR scores of the SP-QAC-$f_d f_q$ seem to deviate from those of the SP-QAC model, even though the differences are small. These observations could be explained by the fact that the signal for personalization from the typed prefix becomes stronger as the prefix length increases, which makes the SP-QAC-$f_p f_q$ and SP-QAC-$f_d f_p$ models considering the factor f_p perform well. Regarding the QAC performance at varying query positions, we can see from Figure 7.1b that the four models have the same performance for the first query and report similar MRR scores at other early positions of a session, e.g., $\#p = 2$ or 3, because not enough search context is available. In addition, the SP-QAC-$f_p f_q$ model performs better than the SP-QAC-$f_d f_q$ and SP-QAC-$f_d f_p$ models at most query positions except for the second query in a session, where the SP-QAC-$f_d f_p$ model presents a bit higher MRR score than the SP-QAC-$f_p f_q$ model. This is because: (1) for the second query, the topic shifts cannot be detected; however, signals for personalization from the typed prefix and the click information still make sense; (2) for the later queries in a session, where a rich search context is provided, valuable information for selective personalization from the typed prefix and the previous queries does indeed help for QAC. In essence, from the results in Figure 7.1, the SP-QAC model which does not consider the factor f_p works worst, by which we infer f_p is the most important factor for selectively personalizing QAC. Similarly, we infer that f_q is more important than f_d.

7.4 Conclusion

In this chapter, we propose a selectively personalized approach for query auto completion. In particular, our model predicts whether a specific prefix should be outweighed on personalization when ranking the query completions. We explore several factors that influence the weight of personalization in a generic personalized QAC model, such as the typed prefix, the clicked documents and the preceding queries in session. We demonstrate that the typed prefix yields the most benefits for weighing personalization in QAC re-ranking and that the preceding queries contributes more than the click information.

This work makes an important step towards unifying prior work on personalized QAC by studying when and how to incorporate personalization in QAC. As to future work, other sources can be explored for investigating how to best personalize query auto completion, e.g., user's dwell time on clicked results and their long-term search history. In addition, it would be interesting to zoom in on particular users to discover whether they stand to benefit from personalization in QAC.

This chapter is the last research chapter of the thesis. The next chapter summarizes the studies presented in this thesis; it summarizes the answers to the research questions formulated in Chapter 1 and gives directions for future work based on the findings in this thesis.

8 Conclusions

In this thesis, we have presented work towards automatically completing users' queries within an information retrieval activity, when only the prefix has been typed in the search box. We approach the problem by trying to answer the question how we can predict users' intended queries and return those early in a list of completion candidates given only few keystrokes as inputs, i.e., the prefix of a query. By now, query auto completion in information retrieval has been well developed; it is commonly integrated into a modern search engine because it can help users avoid spelling mistakes and produce clear query expressions. Where offered, query completion is heavily used by visitors and highly influential on search results, resulting in improved search satisfactions from users.

The four research chapters of this thesis have addressed challenges of query auto completion in the following manner. First, in Chapter 4, we have focused on how to incorporate temporal and user-specific information. In particular, we have analyzed cyclic querying behavior as well as recent trends in query popularity to better predict a query's future popularity. We have also modeled users' interests based on the context from their current search session and their previous sessions, and have proposed a new QAC algorithm to address the limitations of existing approaches. Second, in Chapter 5, we have investigated the limitations of popularity-based query auto completion approaches, where counting the queries follows a strict query matching policy, ignoring the contributions from similar queries. Hence, we have focused on the contributions from so-called homologous queries. In addition, we have paid attention to the semantic similarity between terms when a user formulates queries and have proposed a learn to rank-based query auto completion approach to incorporate these mentioned feature. Next, in Chapter 6, we have turned to a practical issue in query auto completion. A limited number of returned completion candidates does not allow many redundant queries to exist in the list of query completions. Here, we have proposed a greedy query selection approach to return the correct query completions early in a ranked list of candidate completions and at the same time diversify these query completions. Finally, in Chapter 7, we have focused on when to personalize query auto completion, and have developed an approach for selectively personalizing query auto completion. We consider several factors, e.g., the typed prefix, the clicked documents and the preceding queries in session, for weighing personalization in a generic QAC model when being combined together with signal from search popularity to rank query completions.

Below, we provide a more detailed summary of the contributions and results of our

research in §8.1 to answer the research questions presented at the beginning of this thesis. We conclude with an outlook on future research directions in §8.2.

8.1 Main Findings

The first research chapter focuses on the combination of temporal information and a user's personal information to improve the performance of ranking query auto completions. The first research questions we address in Chapter 4 focus on examining the performance of our proposed time-sensitive QAC models, i.e., λ-TS-QAC and λ^*-TS-QAC:

RQ1 As a sanity check, what is the accuracy of query popularity prediction generated by various models?

RQ2 How do our time-sensitive QAC models (λ-TS-QAC and λ^*-TS-QAC) compare against state-of-the-art time-sensitive QAC baselines?

In answering these two research questions, we find that our prediction method, based on the periodicity and on the recent trend of query popularity, can produce accurate predictions of query popularity and perform better in terms of Mean Absolute Error (MAE) and Symmetric Mean Absolute Percentage Error (SMAPE) than other aggregation- and trend-based prediction baselines. Based on predicted query popularity, our proposed time-sensitive QAC model achieves better performance in terms of Mean Reciprocal Rank (MRR) than previous baselines.

After that, we propose a hybrid QAC model λ^*-H-QAC that considers both time-sensitivity and personalization to compare with an n-gram based hybrid model λ^*-H$_G$-QAC. Besides, an extension of λ^*-H-QAC, λ^*-H$'$-QAC, is proposed to deal with long-tail prefixes, i.e., unpopular prefixes, by optimizing the contributions from the predicted query popularity and from the user-specific context. To verify the effectiveness of proposed QAC models, we answer the following research questions:

RQ3 Does λ^*-H-QAC outperform time-sensitive QAC methods, e.g., λ^*-TS-QAC)?

RQ4 How does λ^*-H-QAC compare against personalized QAC method using n-gram based query similarity?

RQ5 How does λ^*-H-QAC compare against λ^*-H$_G$-QAC?

RQ6 How does λ^*-H$'$-QAC compare against λ^*-H-QAC on long-tail prefixes? And on all prefixes?

After incorporating personal information from a particular user into the TS-QAC model, the proposed hybrid model, λ^*-H-QAC, is able to marginally outperform the baselines on query logs at each prefix length. However, despite the additional overhead of scoring similarity between queries, λ^*-H-QAC presents relatively small improvements over proposed λ^*-TS-QAC. In addition, for both the AOL and SvN query logs, λ^*-H-QAC is considerably more effective at longer prefixes as a slightly longer prefix hugely narrows the number of possible completion candidates.

Compared to the personalized QAC scenarios, e.g., G-QAC (n-gram based approach), λ^*-H-QAC significantly outperforms G-QAC in terms of MRR scores at all cases. However, when G-QAC is combined with λ^*-TS-QAC, the combined model, i.e., λ^*-H$_G$-QAC, performs very competitively when compared against λ^*-H-QAC. This appears to

be due to the fact that (1) λ^*-H$_G$-QAC scores the query similarity on a close character level but confronts the sparseness problem, and (2) the number of grams n is artificially fixed, resulting in failure to rank query completions properly. We further extend our model to deal with long-tail prefixes by proposing a modified hybrid QAC model, i.e., λ^*-H$'$-QAC, which receives the highest MRR scores among all QAC models.

In previous work, most of today's QAC models rank candidates by popularity, following a strict query matching policy when counting the queries. Thus, the contributions from so-called homologous queries are ignored. Moreover, today's QAC approaches often ignore semantically related terms. But users are prone to combine semantically related terms when generating queries. To address this shortcoming, based on a learning-based QAC model L2R-U that extracts features from user behavior (Jiang et al., 2014b), we propose several learning to rank-based QAC approaches, where, for the first time, features derived from predicted popularity, homologous queries and semantically related terms are introduced, respectively. In particular, we consider (1) the observed and predicted popularity of query completions, which results in the L2R-UP model; (2) the observed and predicted popularity of homologous queries for a query candidate, which results in the L2R-UPH model; (3) the semantic relatedness of pairs of terms inside a query and pairs of queries inside a session, which results in the L2R-UPS model; and (4) all these newly proposed features, which results in the L2R-ALL model. Regarding these new models, we address the following research questions:

RQ7 Do the features that describe the observed and predicted popularity of a query completion help boost QAC performance without negatively impacting the effectiveness of user behavior related features proposed in (Jiang et al., 2014b)? That is, how does L2R-UP compare against L2R-U?

RQ8 Do semantic features help improve QAC performance? That is, how does L2R-UPS compare against L2R-UP?

RQ9 Do homologous queries help improve QAC performance? That is, how does L2R-UPH compare against L2R-UP?

RQ10 How does L2R-UPS compare against L2R-UPH? What is the performance gain, if any, if all features are added for learning (L2R-ALL)?

RQ11 What are the principal features developed here for a learning to rank based QAC task?

Our experimental analysis reveals that features of semantic relatedness and homologous queries are important and they do indeed help boost QAC performance. In particular, the MRR gains of L2R-UPS over L2R-UP are larger for longer prefixes. This means that features of semantic relatedness are important and do indeed help boost QAC performance. In other words, query terms are not randomly combined when a searcher formulates a query. Semantically close terms or queries are likely to appear in a query or in a session, respectively.

We further extend L2R-UP to examine the contribution from features of homologous queries for the candidate. Across the board, L2R-UPH is found to outperform L2R-UP in terms of MRR and SR@1. L2R-UPH reports an average MRR improvement of nearly 2% over L2R-UP, respectively. Interestingly, the gains in MRR are larger for

shorter prefixes (e.g., $\#p = 1$ or 2). We believe that this is due to the fact that shorter prefixes result in more ambiguous and shorter candidates, leading to a higher probability for query completions to possess homologous queries from which more information can be gleaned.

A central question is which features are relatively useful to a learning to rank-based QAC model? We analyze the relative importance of our newly developed features according to a χ^2 test and find that, generally, semantic relatedness features are more important than those of homologous queries. For instance, the semantic features based on the word2vec score returned by the word2vec model on the query logs, appear to be the most important features.

In Chapter 6 we turn to a practical issue of the query auto completion task in a web search setting: semantically related queries matching the input prefix are often returned together, resulting in a redundancy problem of the list of query candidates. We address this problem of diversifying query auto completion (D-QAC) by proposing a greedy query selection (GQS) model. In particular, we propose a series of greedy query selection (GQS) models, i.e., GQS$_{MPC+AQ}$, GQS$_{MSR+AQ}$, GQS$_{MPC+LQ}$ and GQS$_{MSR+LQ}$, corresponding to a GQS model that first selects the most popular completion and use all previous queries in session as search context, that first selects the most similar completion and use all previous queries in session as search context, that first selects the most popular completion and use only the last preceding query in session as search context and that first selects the most similar completion and use only the last preceding query in session as search context, respectively. We answer the following questions:

RQ12 Do our greedy query selection (GQS) models beat the baselines for diversifying query auto completion task in terms of metrics for QAC ranking (e.g., MRR) and for diversification (e.g., α-nDCG)?

RQ13 How does the choice of selecting the first query to be included in the QAC result list impact the performance in diversified query auto completion of our GQS model?

RQ14 What is the impact on diversified query auto completion performance of our GQS model of the choice of search context, i.e., choosing all previous queries in a session or only the last preceding query?

RQ15 What is the relative D-QAC performance of our QAC models when evaluated using a side-by-side comparison?

RQ16 What is the sensitivity of our GQS model? In particular, how is the performance of our GQS model influenced by, e.g., the number of returned query auto completion candidates, namely a cutoff N, the number of latent features used in BPMF k_f and a trade-off λ controlling the contribution of search popularity and search context when modeling the closeness of query completion to search intent?

We confirm that our greedy query selection approach can indeed remove redundant queries in the original QAC list and boost the QAC performance in the form of returning the final submitted query early and making the final returned list cover more aspects of queries. However, the improvements of our GQS models over the baseline in terms of MRR are limited. Regarding the diversity results, our GQS models report notable improvements. This is probably because for some cases, redundant queries can indeed be

removed from the QAC list by our GQS models; however, these redundant candidates are ranked lower than the final submitted query in the original QAC list and consequently do not affect the reciprocal rank score but do affect the diversity scores.

In addition, our experimental results reveal that the first query selected in our GQS model does impact the QAC ranking performance in the D-QAC tasks. Our GQS model can achieve better D-QAC performance when starting with the most semantically similar query rather than the most popular one. We contribute this to the cases where the user submits similar queries, e.g., an extended query of a preceding one in current session or a repeated query in session.

Regarding the selection of search context used, we concluded that our GQS model using only the last preceding query as search context achieves better D-QAC performance than that using all preceding queries in session as search context. Our additional experiments also reveal that the advantages of our GQS models over the baseline become more prominent when more query completions are initially returned.

Finally, we turn to the question when to incorporate personalization in a generic QAC approach. We assume that the weight of personalization in a hybrid QAC model, which considers both the search popularity and search context when ordering the query completions, can be non-uniformly assigned. Based on a lenient personalized QAC strategy that basically encodes the ranking signal as a trade-off between query popularity and search context, we propose a Selectively Personalizing Query Auto Completion (SP-QAC) model to study such a trade-off. In particular, we predict an effective trade-off in each case based on a regression model, where the typed prefix, the clicked documents and the preceding queries in session are considered for weighing personalization in QAC. The research questions addressed by our study are:

RQ17 Does selective personalization scheme help improve the accuracy of ranking query completions in a generic personalized QAC approach?

RQ18 How is the performance of proposed SP-QAC model under various inputs to the regression model for weighing personalization in a QAC task?

We demonstrate that the typed prefix yields the most benefits for weighing personalization in a QAC model and that the preceding queries contributes more than the click information. This work makes an important step towards unifying prior work on personalized QAC by studying when and how to incorporate personalization in QAC. We will continue to explore other sources for investigating how to best personalize query auto completion, e.g., user's dwell time on clicked results and their long-term search history.

8.2 Future Work

The work presented in this thesis provides insights and algorithms for query auto completion in IR. Beyond the findings and conclusions summarized above, it opens up many important directions for future work. Below, we identify possible follow-up research directions regarding specific areas discussed in the thesis (time-sensitive QAC, learning-based QAC, diversifying QAC) and beyond.

Time-sensitive QAC. As a common query auto completion approach involves re-ranking the top N candidates initially returned by a basic ranker, typically $N = 10$ (Bar-Yossef and Kraus, 2011; Cai et al., 2014b; Shokouhi, 2013), it would be interesting to have a closer look at the top N candidates returned for $N > 10$: how much can we gain from the good candidates that were ranked at lower ranks than 10? A large value of N could bring a possible completion in the QAC list by considering the time related information. Moreover, we aim to transfer our approach to other datasets with long-term query logs, which should help us to benefit from queries with longer periodicity than we have access to in the logs used in our current work. By doing so, time information could bring benefits to more queries by correctly predicting their future popularity.

In addition, we could study a cold-start problem where a user's long-term search logs are unavailable. This problem could be addressed by using the search logs from a group of similar users seen in the training period, because similar users who share common search preferences may have a similar long-term search history (Pan and Chen, 2013). In other words, we can predict a user's interest from a group of similar users.

A further possible step is to model a user's temporal information requests, especially for news search. It has been reported that queries containing temporal search intents account for a notable percentage (Metzler et al., 2009; Nunes et al., 2008) of the whole query stream. The importance of considering temporal aspects in IR and the need for a continuous search for effective temporal IR solutions is evidenced by numerous time-related initiatives and applications (Campos et al., 2014; Diaz et al., 2013; Peetz et al., 2014). Considering temporal information might help generate a better QAC ranking approach after correctly detecting a user's temporal needs, which could be explicitly expressed by terms, like year, month, and day, or implicitly revealed by previous queries or clicked documents in current search session.

Learning-based QAC. Typically, in a learning to rank-based QAC framework, like (Jiang et al., 2014b; Shokouhi, 2013), tens of features are proposed to extract meaningful representations of queries for improving the quality of the ranking of query completions. Hence, efficiency is an important aspect to take into account in a practical QAC setting. Thus, we want to study efficiency aspects of our approaches: parallel processing is likely to boost the learning efficiency of our models on feature extraction, and the addition of more, potentially expensive ways of generating homologous queries or semantic features could produce better QAC rankings. In addition, an online QAC test system should pay attention to the response time, i.e., how fast a user can obtain a QAC list after he or she inputs the prefix. Hence, how to store these query candidates for fast lookups is a key point.

Besides, in addition to the proposed features of semantic relatedness and of homologous queries, we aim to develop new features that can capture a deep understanding of candidate query completions. For instance, we can resort to a high resolution query log, where users' detailed interaction data at each keystroke is recorded. As shown in (Li et al., 2015), from this rich interaction data behavioral features of a user's implicit feedback can be extracted to indicate the probability or preference of user's intended query. It would be interesting to consider user behavior like skipping or viewing the QAC list (Zhang et al., 2015) as well as the typing speed (Mitra et al., 2014), which should enable more accurate models for predicting a user's intended query.

Diversifying QAC. In our current solution to the D-QAC problem, the ground truth data of query relevance to aspects is automatically generated from clickthrough data. Like (Vallet, 2011; Vallet and Castells, 2011), we plan to move our models to other datasets, where the ground truth of query aspects is generated by real humans rather than automatically generated. It is interesting to see the difference between the human judgements and the automatic estimation scenario on the relevance of query to aspects. Like (Chapelle et al., 2012; Vallet, 2011; Vallet et al., 2010) we could run a crowdsourcing experiment to implement a side-by-side comparison and give an outcome of pairwise comparisons between various QAC models rather than implementing a questionnaire in a laboratory setting as used in Chapter 6.

In addition, it would be interesting to consider other scenarios for dealing with the query cold-start problem as this may be helpful to obtain the correct query distribution over aspects. Semantic similarity between query terms or queries, which is effective for finding close neighbors of queries (Han et al., 2013; Jones et al., 2006), can be helpful to find similar query terms or queries, by which we can assign the aspect label to those unknown queries.

Other follow-ups. Looking ahead more broadly, one possible direction of development is to consider the entities in a query. It has been reported that a remarkable proportion of queries in query logs involve actions on entities (Guo et al., 2009b; Lin et al., 2012), calling for an automatic approach to identifying entity-related queries. One of the key challenges in considering entity information for query auto completion is the correct detection and recognition of entities, which would involve the segmentation of queries, the classification of entities as well as the mapping of entities to a pre-known entity base. From an algorithmic side, this could result in methods for promoting entity-related queries that are close to user's search context and possible to be issued. Similar methods have been developed in the web search community (Reinanda et al., 2015), but need to be adapted to the application in query auto completion.

Another direction is to develop QAC models for mobile search. The use of search engines on mobile devices, has experienced a rapid growth in past few years (Church and Oliver, 2011). So far, QAC in the specific setting of mobile search has not been well studied. Text input is relatively slower and clumsier than in traditional desktop search because of a smaller screen of mobile devices. Hence, assisting users to formulate their queries merits special attention. Previous approaches to QAC do not fully exploit the spatiotemporal information of query candidates, such as the query time, the direction in which a user moves or the distance from user to a candidate query completion. Such spatiotemporal information could be particularly important to QAC for mobile search, especially for location search on mobile devices.

Bibliography

E. Agichtein, C. Castillo, D. Donato, A. Gionis, and G. Mishne. Finding high-quality content in social media. In *WSDM '08*, pages 183–194, New York, NY, USA, 2008. ACM. (Cited on page 82.)

R. Agrawal, S. Gollapudi, A. Halverson, and S. Ieong. Diversifying search results. In *WSDM '09*, pages 5–14, New York, NY, USA, 2009. ACM. (Cited on page 36.)

A. Amin, M. Hildebrand, J. van Ossenbruggen, V. Evers, and l. Hardman. Organizing suggestions in autocompletion interfaces. In *ECIR '09*, pages 521–529, 2009. (Cited on page 29.)

K. Bache, D. Newman, and P. Smyth. Text-based measures of document diversity. In *KDD '13*, pages 23–31, New York, NY, USA, 2013. ACM. (Cited on page 3.)

Z. Bar-Yossef and N. Kraus. Context-sensitive query auto-completion. In *WWW '11*, pages 107–116, New York, NY, USA, 2011. ACM. (Cited on pages 1, 2, 13, 15, 18, 19, 20, 23, 28, 31, 33, 34, 39, 43, 47, 51, 54, 74, 75, 76, 85, 86, 95, 96, 97, 115, 116, 117, and 128.)

H. Bast and I. Weber. Type less, find more: Fast autocompletion search with a succinct index. In *SIGIR '06*, pages 364–371, New York, NY, USA, 2006. ACM. (Cited on page 1.)

H. Bast, A. Chitea, F. Suchanek, and I. Weber. Ester: Efficient search on text, entities, and relations. In *SIGIR '07*, pages 671–678, 2007. (Cited on page 25.)

S. M. Beitzel, E. C. Jensen, O. Frieder, A. Chowdhury, and G. Pass. Surrogate scoring for improved metasearch precision. In *SIGIR '05*, pages 583–584, 2005. (Cited on page 28.)

P. N. Bennett, R. W. White, W. Chu, S. T. Dumais, P. Bailey, F. Borisyuk, and X. Cui. Modeling the impact of short- and long-term behavior on search personalization. In *SIGIR '12*, pages 185–194, New York, NY, USA, 2012. ACM. (Cited on pages 18, 22, 52, and 99.)

D. Bollegala, Y. Matsuo, and M. Ishizuka. Measuring semantic similarity between words using web search engines. In *WWW '07*, pages 757–766, New York, NY, USA, 2007. ACM. (Cited on page 92.)

C. J. Burges, K. M. Svore, P. N. Bennett, A. Pastusiak, and Q. Wu. Learning to rank using an ensemble of lambda-gradient models. *J. Mach. Learn. Res.*, 14:25–35, 2011. (Cited on pages 22, 67, 68, and 77.)

F. Cai and M. de Rijke. Time-aware personalized query auto completion. In *DIR '15*, page 12, 2015a. (Cited on page 11.)

F. Cai and M. de Rijke. Personalized web search based on bayesian probabilistic matrix factorization. In *RuSSIR '15*, pages 39–40, 2015b. (Cited on page 11.)

F. Cai and M. de Rijke. Learning from homologous queries and semantically related terms for query auto completion. *Information Processing and Management*, 2016a. To appear. (Cited on pages 10, 15, 23, 26, 33, and 34.)

F. Cai and M. de Rijke. Query auto completion in information retrieval: A survey. *Foundations and Trends in Information Retrieval*, 2016b. Submitted. (Cited on page 10.)

F. Cai and M. de Rijke. Selectively personalizing query auto completion. In *SIGIR '16*. ACM, 2016c. To appear. (Cited on page 10.)

F. Cai, S. Liang, and M. de Rijke. Personalized document re-ranking based on bayesian probabilistic matrix factorization. In *SIGIR '14*, pages 835–838, New York, NY, USA, 2014a. ACM. (Cited on pages 11, 46, 77, 82, and 95.)

F. Cai, S. Liang, and M. de Rijke. Time-sensitive personalized query auto-completion. In *CIKM '14*, pages 1599–1608, New York, NY, USA, 2014b. ACM. (Cited on pages 10, 15, 17, 18, 20, 21, 23, 28, 31, 33, 34, 76, 77, 83, 97, 100, 116, 117, 119, and 128.)

F. Cai, S. Liang, and M. de Rijke. Prefix-adaptive and time-sensitive personalized query auto completion. *IEEE Transaction on Knowledge and Data Engineering*, 2016a. Accepted subject to minor revisions. (Cited on pages 10 and 23.)

F. Cai, R. Reinanda, and M. de Rijke. Diversifying query auto-completion. *ACM Transactions on Information Systems*, 2016b. To appear. (Cited on pages 10, 30, 34, 36, and 63.)

F. Cai, S. Wang, and M. de Rijke. Behavior-based personalization in web search. *Journal of the Association for Information Science and Technology*, 2016c. To appear. (Cited on page 11.)

R. Campos, G. Dias, A. M. Jorge, and A. Jatowt. Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*, 47(2):15:1–15:41, 2014. (Cited on page 128.)

H. Cao, D. Jiang, J. Pei, Q. He, Z. Liao, E. Chen, and H. Li. Context-aware query suggestion by mining click-through and session data. In *KDD '08*, pages 875–883, 2008. (Cited on page 22.)

J. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR '98*, pages 335–336, New York, NY, USA, 1998. ACM. (Cited on pages 3, 96, and 97.)

O. Chapelle and Y. Zhang. A dynamic bayesian network click model for web search ranking. In *WWW '09*,

pages 1–10, 2009. (Cited on page 24.)
O. Chapelle, D. Metlzer, Y. Zhang, and P. Grinspan. Expected reciprocal rank for graded relevance. In *CIKM '09*, pages 621–630, New York, NY, USA, 2009. ACM. (Cited on pages 36 and 94.)
O. Chapelle, T. Joachims, F. Radlinski, and Y. Yue. Large-scale validation and analysis of interleaved search evaluation. *ACM Trans. Inf. Syst.*, 30(1):6:1–6:41, 2012. (Cited on pages 98, 99, 108, and 129.)
C. Chatfield. *The Analysis of Time Series: An Introduction*. Chapman and Hall, New York, 2004. (Cited on pages 44 and 69.)
S. Chaudhuri and R. Kaushik. Extending autocompletion to tolerate errors. In *SIGMOD '09*, pages 707–718, New York, NY, USA, 2009. ACM. (Cited on page 26.)
S. Chien and N. Immorlica. Semantic similarity between search engine queries using temporal correlation. In *WWW '05*, pages 2–11, New York, NY, USA, 2005. ACM. (Cited on pages 18, 73, and 92.)
A. Chuklin, I. Markov, and M. de Rijke. An introduction to click models for web search: Sigir 2015 tutorial. In *SIGIR '15*, pages 1113–1115, 2015a. (Cited on page 30.)
A. Chuklin, I. Markov, and M. de Rijke. *Click Models for Web Search*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, San Rafael, CA, USA, August 2015b. (Cited on pages 24 and 30.)
K. Church and N. Oliver. Understanding mobile web and mobile search use in today's dynamic mobile landscape. In *MobileHCI '11*, pages 67–76, 2011. (Cited on page 129.)
C. L. Clarke, M. Kolla, G. V. Cormack, O. Vechtomova, A. Ashkan, S. Büttcher, and I. MacKinnon. Novelty and diversity in information retrieval evaluation. In *SIGIR '08*, pages 659–666, New York, NY, USA, 2008. ACM. (Cited on page 36.)
C. L. Clarke, M. Kolla, and O. Vechtomova. An effectiveness measure for ambiguous and underspecified queries. In *ICTIR '09*, pages 188–199, Berlin, Heidelberg, 2009. Springer-Verlag. (Cited on page 36.)
C. L. A. Clarke, E. Agichtein, S. Dumais, and R. W. White. The influence of caption features on clickthrough patterns in web search. In *SIGIR '07*, pages 135–142, 2007. (Cited on page 29.)
K. Collins-Thompson, P. N. Bennett, R. W. White, S. de la Chica, and D. Sontag. Personalizing web search results by reading level. In *CIKM '11*, pages 403–412, 2011. (Cited on page 18.)
K. Collins-Thompson, P. Bennett, C. L. A. Clarke, and E. M. Voorhees. Trec 2013 web track overview. In *TREC '13*, pages 1–15, Berlin, Heidelberg, 2013. Springer-Verlag. (Cited on page 36.)
N. Craswell, O. Zoeter, M. Taylor, and B. Ramsey. An experimental comparison of click position-bias models. In *WSDM '08*, pages 87–94, 2008. (Cited on page 24.)
N. Craswell, R. Jones, G. Dupret, and E. Viegas, editors. *WSCD '09: Proceedings 2009 Workshop on Web Search Click Data*, 2009. ACM. (Cited on pages 34, 49, and 75.)
W. B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*, volume 1. Pearson Education, Inc., 2015. (Cited on page 1.)
S. Cucerzan. Large-scale named entity disambiguation based on wikipedia data. In *Proceedings of EMNLP-CoNLL 2007*, pages 708–716, June 2007. (Cited on page 26.)
V. Dang and W. B. Croft. Diversity by proportionality: An election-based approach to search result diversification. In *SIGIR '12*, pages 65–74, New York, NY, USA, 2012. ACM. (Cited on page 3.)
A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Stat. Soc., Series B*, 39(1):1–38, 1977. (Cited on page 95.)
G. Di Santo, R. McCreadie, C. Macdonald, and I. Ounis. Comparing approaches for query autocompletion. In *SIGIR '15*, pages 775–778, 2015. (Cited on page 14.)
F. Diaz, S. Dumais, M. Efron, K. Radinsky, M. de Rijke, and M. Shokouhi. Sigir 2013 workshop on time-aware information access (#taia2013). In *SIGIR '13: 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, July 2013. (Cited on page 128.)
Z. Dou, R. Song, and J.-R. Wen. A large-scale evaluation and analysis of personalized search strategies. In *WWW '07*, pages 581–590, 2007. (Cited on pages 19 and 115.)
H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*, pages 117–126, 2011a. (Cited on page 26.)
H. Duan and B.-J. P. Hsu. Online spelling correction for query completion. In *WWW '11*, pages 117–126, 2011b. (Cited on page 34.)
G. E. Dupret and B. Piwowarski. A user browsing model to predict search engine click data from past observations. In *SIGIR '08*, pages 331–338, 2008. (Cited on page 24.)
S. Fox, K. Karnawat, M. Mydland, S. Dumais, and T. White. Evaluating implicit measures to improve web search. *ACM Trans. Inf. Syst.*, 23(2):147–168, 2005. (Cited on page 117.)
G. Francès, X. Bai, B. B. Cambazoglu, and R. Baeza-Yates. Improving the efficiency of multi-site web search engines. In *WSDM '14*, pages 3–12, 2014. (Cited on page 25.)

J. a. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia. A survey on concept drift adaptation. *ACM Comput. Surv.*, 46(4):44:1–44:37, March 2014. (Cited on pages 49 and 75.)

N. G. Golbandi, L. K. Katzir, Y. K. Koren, and R. L. Lempel. Expediting search trend detection via prediction of query counts. In *WSDM '13*, pages 295–304, 2013. (Cited on pages 17 and 18.)

P. Goodwin. The holt-winters approach to exponential smoothing: 50 years old and going strong. *Foresight: The International Journal of Applied Forecasting*, 1(19):30–33, 2010. (Cited on page 16.)

L. A. Granka, T. Joachims, and G. Gay. Eye-tracking analysis of user behavior in www search. In *SIGIR '04*, pages 478–479, 2004. (Cited on page 24.)

F. Guo, C. Liu, and Y. M. Wang. Efficient multiple-click models in web search. In *WSDM '09*, pages 124–131, 2009a. (Cited on page 30.)

J. Guo, G. Xu, X. Cheng, and H. Li. Named entity recognition in query. In *SIGIR '09*, pages 267–274, New York, NY, USA, 2009b. ACM. (Cited on page 129.)

L. Han, A. L. Kashyap, T. Finin, J. Mayfield, and J. Weese. UMBC EBIQUITY-CORE: Semantic textual similarity systems. In *2nd Joint Conference on Lexical and Computational Semantics*. ACL, 2013. (Cited on pages 71 and 129.)

J. He, E. Meij, and M. de Rijke. Result diversification based on query-specific cluster ranking. *J. Am. Soc. Inf. Sci. Technol.*, 62(3):550–571, 2011. (Cited on pages 96 and 97.)

Q. He, D. Jiang, Z. Liao, S. C. H. Hoi, K. Chang, E.-P. Lim, and H. Li. Web query recommendation via sequential query prediction. In *ICDE '09*, pages 1443–1454, 2009. (Cited on page 23.)

K. Hofmann, B. Mitra, F. Radlinski, and M. Shokouhi. An eye-tracking study of user interactions with query auto completion. In *CIKM '14*, pages 549–558, New York, NY, USA, 2014. ACM. (Cited on pages 30 and 36.)

C. C. Holt. Forecasting seasonals and trends by exponentially weighted moving averages. *International Journal of Forecasting*, 20(1):5–10, 2004. (Cited on pages 16 and 17.)

B.-J. P. Hsu and G. Ottaviano. Space-efficient data structures for top-k completion. In *WWW '13*, pages 583–594, 2013. (Cited on page 25.)

B. Huurnink, L. Hollink, W. van den Heuvel, and M. de Rijke. Search behavior of media professionals at an audiovisual archive: A transaction log analysis. *J. Am. Soc. Inf. Sci. Techn.*, 61(6):1180–1197, June 2010. (Cited on pages 35, 49, and 50.)

A. Jain and G. Mishne. Organizing query completions for web search. In *CIKM '10*, pages 1169–1178, 2010. (Cited on pages 29 and 30.)

K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4): 422–446, 2002. (Cited on page 36.)

S. Ji, G. Li, C. Li, and J. Feng. Efficient interactive fuzzy keyword search. In *WWW '09*, pages 371–380, 2009. (Cited on page 25.)

D. Jiang, K. W.-T. Leung, and W. Ng. Context-aware search personalization with concept preference. In *CIKM '11*, pages 563–572, 2011. (Cited on page 18.)

J. Jiang, D. He, and J. Allan. Searching, browsing, and clicking in a search session: Changes in user behavior by task and over time. In *SIGIR '14*, pages 607–616, 2014a. (Cited on page 118.)

J.-Y. Jiang, Y.-Y. Ke, P.-Y. Chien, and P.-J. Cheng. Learning user reformulation behavior for query auto-completion. In *SIGIR '14*, pages 445–454, New York, NY, USA, 2014b. ACM. (Cited on pages 5, 15, 21, 22, 23, 26, 33, 34, 36, 66, 67, 73, 74, 75, 76, 77, 82, 83, 97, 100, 117, 119, 125, and 128.)

T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02*, pages 133–142, New York, NY, USA, 2002. ACM. (Cited on page 93.)

T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *SIGIR '05*, pages 154–161, 2005. (Cited on page 29.)

H. Joho and J. M. Jose. A comparative study of the effectiveness of search result presentation on the web. In *ECIR '06*, pages 302–313, 2006. (Cited on page 29.)

H. Joho and J. M. Jose. Effectiveness of additional representations for the search result presentation on the web. *Information Processing and Management*, 44(1):226–241, 2008. (Cited on page 29.)

R. Jones and K. L. Klinkner. Beyond the session timeout: Automatic hierarchical segmentation of search topics in query logs. In *CIKM '08*, pages 699–708, 2008. (Cited on page 34.)

R. Jones, B. Rey, O. Madani, and W. Greiner. Generating query substitutions. In *WWW '06*, pages 387–396, New York, NY, USA, 2006. ACM. (Cited on pages 58, 72, and 129.)

D. Kastrinakis and Y. Tzitzikas. Advancing search query autocompletion services with more and better suggestions. In *ICWE'10*, pages 35–49, 2010. (Cited on page 25.)

G. Kazai, J. Kamps, M. Koolen, and N. Milic-Frayling. Crowdsourcing for book search evaluation: Impact of hit design on comparative system ranking. In *SIGIR '11*, pages 205–214, 2011. (Cited on page 28.)

Y. Kim, A. Hassan, R. W. White, and I. Zitouni. Modeling dwell time to predict click-level satisfaction. In *WSDM '14*, pages 193–202, 2014. (Cited on page 117.)

A. Kulkarni, J. Teevan, K. M. Svore, and S. T. Dumais. Understanding temporal query dynamics. In *WSDM '11*, pages 167–176, 2011. (Cited on page 17.)

G. Li, S. Ji, C. Li, and J. Feng. Efficient fuzzy full-text type-ahead search. *The VLDB Journal*, 20(4):617–640, 2011. (Cited on page 25.)

L. Li, H. Deng, A. Dong, Y. Chang, H. Zha, and R. Baeza-Yates. Analyzing user's sequential behavior in query auto-completion via markov processes. In *SIGIR '15*, pages 123–132, New York, NY, USA, 2015. ACM. (Cited on pages 15, 19, 20, 29, 31, 34, and 128.)

Y. Li, A. Dong, H. Wang, H. Deng, Y. Chang, and C. Zhai. A two-dimensional click model for query auto-completion. In *SIGIR '14*, pages 455–464, New York, NY, USA, 2014. ACM. (Cited on pages 15, 22, 23, 29, 30, 31, and 34.)

S. Liang, F. Cai, Z. Ren, and M. de Rijke. Efficient structured learning for personalized diversification. Manuscript submitted to IEEE TKDE, 2015. (Cited on page 11.)

Z. Liao, D. Jiang, E. Chen, J. Pei, H. Cao, and H. Li. Mining concept sequences from large-scale search logs for context-aware query suggestion. *ACM Trans. Intell. Syst. Technol.*, 3(1):Article 17, 2011. (Cited on pages 19, 23, and 33.)

T. Lin, P. Pantel, M. Gamon, A. Kannan, and A. Fuxman. Active objects: Actions for entity-centric search. In *WWW '12*, pages 589–598, New York, NY, USA, 2012. ACM. (Cited on page 129.)

W. Litwin, R. Mokadem, P. Rigaux, and T. Schwarz. Fast ngram-based string search over data encoded using algebraic signatures. In *VLDB '07*, pages 207–218, 2007. (Cited on page 53.)

C. Liu, F. Guo, and C. Faloutsos. Bayesian browsing model: Exact inference of document relevance from petabyte-scale data. *ACM Transactions on Knowledge Discovery from Data*, 4(4):19:1–19:26, 2010a. (Cited on page 24.)

C. Liu, R. W. White, and S. Dumais. Understanding web browsing behaviors through weibull analysis of dwell time. In *SIGIR '10*, pages 379–386, 2010b. (Cited on page 18.)

N. Liu, J. Yan, S. Yan, W. Fan, and Z. Chen. Web query prediction by unifying model. In *ICDM '08*, pages 437–441, 2008. (Cited on page 18.)

T.-Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3): 225–331, 2003. (Cited on pages 20 and 26.)

C. Lucchese, S. Orlando, R. Perego, F. Silvestri, and G. Tolomei. Identifying task-based sessions in search engine query logs. In *WSDM '11*, pages 277–286, 2011. (Cited on page 34.)

H. Ma, H. Yang, I. King, and M. R. Lyu. Learning latent semantic relations from clickthrough data for query suggestion. In *CIKM '08*, pages 709–718, 2008. (Cited on page 33.)

G. Marchionini and R. White. Find what you need, understand what you find. *International Journal of Human [# x02013] Computer Interaction*, 23(3):205–237, 2007. (Cited on page 28.)

D. Matani. An o(k log n) algorithm for prefix based ranked autocomplete. `http://www.dhruvbird.com/autocomplete.pdf`, 2011. (Cited on page 25.)

N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *WSDM '11*, pages 25–34, 2011. (Cited on page 18.)

Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08*, pages 469–478, 2008a. (Cited on page 22.)

Q. Mei, D. Zhou, and K. Church. Query suggestion using hitting time. In *CIKM '08*, pages 469–478, 2008b. (Cited on page 33.)

D. Metzler, R. Jones, F. Peng, and R. Zhang. Improving search relevance for implicitly temporal queries. In *SIGIR '09*, pages 700–701, New York, NY, USA, 2009. ACM. (Cited on page 128.)

V. Michail, M. Christopher, V. Zografoula, and G. Dimitrios. Identifying similarities periodicities and bursts for online search queries. In *SIGMOD '04*, pages 131–142, 2004. (Cited on page 17.)

T. Mikolov, K. Chen, G. S. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at ICLR*, pages 1–13, Cambridge, Massachusetts, 2013a. MIT Press. (Cited on pages 72 and 93.)

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *NIPS 26*, pages 3111–3119, Cambridge, Massachusetts, 2013b. MIT Press. (Cited on pages 72, 91, 93, 116, and 118.)

B. Mitra. Exploring session context using distributed representations of queries and reformulations. In *SIGIR '15*, pages 3–12, New York, NY, USA, 2015. ACM. (Cited on pages 15, 21, and 23.)

B. Mitra and N. Craswell. Query auto-completion for rare prefixes. In *CIKM '15*, pages 1755–1758, 2015. (Cited on page 23.)

B. Mitra, M. Shokouhi, F. Radlinski, and K. Hofmann. On user interactions with query auto-completion. In *SIGIR '14*, pages 1055–1058, New York, NY, USA, 2014. ACM. (Cited on pages 24, 30, and 128.)

M. R. Morris, J. Teevan, and S. Bush. Enhancing collaborative web search with personalization: Groupization, smart splitting, and group hit-highlighting. In *CSCW '08*, pages 481–484, 2008. (Cited on page 28.)

R. M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, September 1993. (Cited on page 95.)

S. Nunes, C. Ribeiro, and G. David. Use of temporal expressions in web search. In *ECIR '08*, pages 580–584, Berlin, Heidelberg, 2008. Springer-Verlag. (Cited on page 128.)

W. Pan and L. Chen. Gbpr: Group preference based bayesian personalized ranking for one-class collaborative filtering. In *IJCAI '13*, pages 2691–2697, 2013. (Cited on page 128.)

G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *InfoScale '06*, pages 1–7, New York, NY, USA, 2006. ACM. (Cited on pages 34, 39, 49, 75, and 85.)

M.-H. Peetz, E. Meij, and M. de Rijke. Using temporal bursts for query modeling. *Information Retrieval Journal*, 17(1):74–108, February 2014. doi: 10.1007/s10791-013-9227-2. (Cited on page 128.)

F. Radlinski and S. Dumais. Improving personalized web search using result diversification. In *SIGIR '06*, pages 691–692, New York, NY, USA, 2006. ACM. (Cited on page 3.)

R. Reinanda, E. Meij, and M. de Rijke. Mining, ranking and recommending entity aspects. In *SIGIR '15*, pages 263–272, New York, NY, USA, 2015. ACM. (Cited on page 129.)

M. Richardson, E. Dominowska, and R. Ragno. Predicting clicks: Estimating the click-through rate for new ads. In *WWW '07*, pages 521–530, 2007. (Cited on page 24.)

R. Salakhutdinov and A. Mnih. Bayesian probabilistic matrix factorization using markov chain monte carlo. In *ICML '08*, pages 880–887, New York, NY, USA, 2008a. ACM. (Cited on pages 94 and 95.)

R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *NIPS 20*, pages 1–8, Cambridge, Massachusetts, 2008b. MIT Press. (Cited on page 95.)

R. L. T. Santos, C. Macdonald, and I. Ounis. Learning to rank query suggestions for adhoc and diversity search. *Inf. Retr.*, 16:429–451, 2013. (Cited on page 19.)

X. Shen, B. Tan, and C. Zhai. Context-sensitive information retrieval using implicit feedback. In *SIGIR '05*, pages 43–50, 2005. (Cited on page 18.)

Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *WWW '14*, pages 373–374, 2014. (Cited on page 23.)

M. Shokouhi. Detecting seasonal queries by time-series analysis. In *SIGIR '11*, pages 1171–1172, New York, NY, USA, 2011. ACM. (Cited on pages 2, 4, 16, and 18.)

M. Shokouhi. Learning to personalize query auto-completion. In *SIGIR '13*, pages 103–112, New York, NY, USA, 2013. ACM. (Cited on pages 15, 19, 21, 22, 23, 26, 28, 33, 34, 65, 68, 76, 77, 97, 100, 117, 119, and 128.)

M. Shokouhi and K. Radinsky. Time-sensitive query auto-completion. In *SIGIR '12*, pages 601–610, New York, NY, USA, 2012. ACM. (Cited on pages 2, 15, 16, 17, 18, 20, 23, 28, 34, 39, 43, and 54.)

D. Sontag, K. Collins-Thompson, P. N. Bennett, R. W. White, S. Dumais, and B. Billerbeck. Probabilistic models for personalizing web search. In *WSDM '12*, pages 433–442, 2012. (Cited on page 19.)

A. Strizhevskaya, A. Baytin, I. Galinskaya, and P. Serdyukov. Actualization of query suggestions using query logs. In *WWW '12*, pages 611–612, New York, NY, USA, 2012. ACM. (Cited on pages 2, 17, and 39.)

B. Tan, X. Shen, and C. Zhai. Mining long-term search history to improve search accuracy. In *KDD '06*, pages 718–723, 2006. (Cited on page 18.)

J. Teevan, S. T. Dumais, and D. J. Liebling. To personalize or not to personalize: Modeling queries with variation in user intent. In *SIGIR '08*, pages 163–170, 2008. (Cited on page 115.)

J. Teevan, D. J. Liebling, and G. Ravichandran Geetha. Understanding and predicting personal navigation. In *WSDM '11*, pages 85–94, 2011. (Cited on page 22.)

P. Thomas and D. Hawking. Evaluation by comparing result sets in context. In *CIKM '06*, pages 94–101, New York, NY, USA, 2006. ACM. (Cited on page 98.)

Y. Ustinovskiy and P. Serdyukov. Personalization of web-search using short-term browsing context. In *CIKM '13*, pages 1979–1988, 2013. (Cited on page 18.)

D. Vallet. Crowdsourced evaluation of personalization and diversification techniques in web search. In *CIR '11 Workshop (SIGIR 2011)*, pages 1–6, New York, NY, USA, 2011. ACM. (Cited on pages 98 and 129.)

D. Vallet and P. Castells. On diversifying and personalizing web search. In *SIGIR '11*, pages 1157–1158, New York, NY, USA, 2011. ACM. (Cited on pages 98 and 129.)

D. Vallet, I. Cantador, and J. M. Jose. Personalizing web search with folksonomy-based user and document profiles. In *ECIR'2010*, pages 420–431. Springer-Verlag, 2010. (Cited on page 129.)

S. Whiting and J. M. Jose. Recent and robust query auto-completion. In *WWW '14*, pages 971–982, New York,

NY, USA, 2014. ACM. (Cited on pages 4, 15, 17, 18, 21, 23, 34, 43, 44, 51, 68, 74, and 75.)

S. Whiting, J. McMinn, and J. M. Jose. Exploring real-time temporal query auto-completion. In *Proceedings of the 12th Dutch-Belgian Information Retrieval workshop*, pages 1–4, 2013. (Cited on pages 16 and 18.)

B. Xiang, D. Jiang, J. Pei, X. Sun, E. Chen, and H. Li. Context-aware ranking in web search. In *SIGIR '10*, pages 451–458, 2010. (Cited on page 22.)

C. Xiao, J. Qin, W. Wang, Y. Ishikawa, K. Tsuda, and K. Sadakane. Efficient error-tolerant query autocompletion. *Proceedings of the VLDB Endowment*, 6(6):373–384, 2013. (Cited on page 25.)

Y. Yue, R. Patel, and H. Roehrig. Beyond position bias: Examining result attractiveness as a source of presentation bias in clickthrough data. In *WWW '10*, pages 1011–1018, 2010. (Cited on page 30.)

C. X. Zhai, W. W. Cohen, and J. Lafferty. Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval. In *SIGIR '03*, pages 10–17, New York, NY, USA, 2003. ACM. (Cited on page 3.)

A. Zhang, A. Goyal, W. Kong, H. Deng, A. Dong, Y. Chang, C. A. Gunter, and J. Han. adaqac: Adaptive query auto-completion via implicit negative feedback. In *SIGIR '15*, pages 143–152, New York, NY, USA, 2015. ACM. (Cited on pages 19, 20, 24, 29, 30, 31, 34, and 128.)

W. Zhang, J. Wang, B. Chen, and X. Zhao. To personalize or not: A risk management perspective. In *RecSys '13*, pages 229–236, 2013. (Cited on page 115.)

Y. Zhang, W. Chen, D. Wang, and Q. Yang. User-click modeling for understanding and predicting search-behavior. In *KDD '11*, pages 1388–1396, 2011. (Cited on page 24.)

Z. A. Zhu, W. Chen, T. Minka, C. Zhu, and Z. Chen. A novel click model and its applications to online advertising. In *WSDM '10*, pages 321–330, 2010. (Cited on page 24.)

Summary

Query auto completion is an important feature embedded into today's search engines. It can help users formulate queries which other people have searched for when he/she finishes typing the query prefix. Today's most sophisticated query auto completion approaches are based on the collected query logs to provide the best possible queries for each searcher's input.

In this thesis, we develop new query auto completion methods for information retrieval. First, we consider the information of both time and user to propose a time-sensitive personalized query auto completion approach. In previous work, these two sources of information have been developed separately. We bring them together and pay special attention to long-tail prefixes. Second, based on a learning-to-rank framework, we propose to extract features originating from so-called homologous queries and from the semantic similarity of terms, which allow the contributions from similar queries and from semantic relatedness to be used for query auto completion.

In addition, we study the problem of query auto completion diversification, where we aim to diversify aspect-level query intents of query completions. This task has not been studied before. Given that only a limited number of query completions can be returned to users of a search engine, it is important to remove redundant queries and improve user satisfaction by finding an acceptable query. Finally, we conduct an investigation on when to personalize query auto completion by proposing a selectively personalizing query auto completion approach, where the weight of personalization in a query auto completion model is selectively assigned based on the search context in session.

The experimental results in this thesis indicate that our proposed query auto completion approaches can improve the ranking performance of query completions in terms of well-known metrics, like Mean Reciprocal Rank. The unique insights and interesting findings in this thesis may be used to help search engine designers to improve the satisfaction of search engine user by providing high quality query completions.

Samenvatting

Automatische aanvulling van zoekopdrachten (*query auto completion*) is een belangrijk kenmerken van hedendaagse zoekmachines. Het helpt gebruikers met het formuleren van zoekopdrachten waar andere gebruikers op hebben gezocht nadat zij zijn gestopt met het typen van een gedeeltelijke zoekopdracht. De meest geavanceerde huidige methodes voor automatische aanvulling van zoekopdrachten maken gebruik van eerder verzamelde zoekopdrachten om zo de best mogelijke zoekopdrachten aan te bieden voor de invoer van iedere gebruiker.

In dit proefschrift ontwikkelen we nieuwe methoden voor het automatisch aanvullen van zoekopdrachten. Ten eerste stellen we een tijdsgevoelige en gepersonaliseerde aanpak voor, die gebruik maakt van temporele informatie en informatie over de gebruiker. Dit in tegenstelling tot eerder werk, waarin deze twee bronnen van informatie los van elkaar worden ontwikkeld. We breiden deze aanpak uit om specifiek geschikt te zijn voor zeldzame gedeeltelijke zoekopdrachten. Ten tweede, in een raamwerk van machine leertechnieken voor het ordenen van resultaten, stellen we voor om eigenschappen te extraheren uit zogenaamde homologe zoekopdrachten en uit de semantische gelijkenis van termen. Dit maakt het mogelijk om gebruik te maken van vergelijkbare zoekopdrachten bij het automatisch aanvullen van zoekopdrachten.

Vervolgens behandelen we in dit proefschrift het diversificeren van automatisch aangevulde zoektermen. Hierbij richten we ons op het diversificeren op aspect-niveau, een nieuwe taak die niet eerder is onderzocht. Omdat er maar een beperkt aantal resultaten wordt getoond, kan het voor de gebruiker gunstig zijn om overlappende resultaten achterwege te laten. Tot slot onderzoeken we wanneer zoektermen op gepersonaliseerde wijze moeten worden aangevuld. We introduceren een selectieve gepersonaliseerde aanpak, waarbij de mate van personalisering selectief gewogen wordt op basis van de context in de zoeksessie van de gebruiker.

De experimentele resultaten in dit proefschrift tonen aan dat de door ons voorgestelde methode beter in staat is zoekopdrachten aan te vullen, gemeten met standaard-metrieken zoals Mean Reciprocal Rank. De unieke inzichten en interessante bevindingen uit dit proefschrift kunnen door ontwikkelaars van zoekmachines ingezet worden om de gebruikerservaring te verbeteren door vaker de juiste aanvulling van een zoekopdracht voor te stellen.

SIKS Dissertation Series

1998

1 Johan van den Akker (CWI) *DEGAS: An Active, Temporal Database of Autonomous Objects*
2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*
3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations*
4 Dennis Breuker (UM) *Memory versus Search in Games*
5 E. W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 Mark Sloof (VUA) *Physiology of Quality Change Modelling: Automated modelling of*
2 Rob Potharst (EUR) *Classification using decision trees and neural nets*
3 Don Beal (UM) *The Nature of Minimax Search*
4 Jacques Penders (UM) *The practical Art of Moving Physical Objects*
5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven*
6 Niek J. E. Wijngaards (VUA) *Re-design of compositional systems*
7 David Spelt (UT) *Verification support for object database design*
8 Jacques H. J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism*

2000

1 Frank Niessink (VUA) *Perspectives on Improving Software Maintenance*
2 Koen Holtman (TUe) *Prototyping of CMS Storage Management*
3 Carolien M. T. Metselaar (UvA) *Sociaal-organisatorische gevolgen van kennistechnologie*
4 Geert de Haan (VUA) *ETAG, A Formal Model of Competence Knowledge for User Interface*
5 Ruud van der Pol (UM) *Knowledge-based Query Formulation in Information Retrieval*
6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*
7 Niels Peek (UU) *Decision-theoretic Planning of Clinical Patient Management*
8 Veerle Coupé (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*
9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*
10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*
11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*
2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*
3 Maarten van Someren (UvA) *Learning as problem solving*
4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*
5 Jacco van Ossenbruggen (VUA) *Processing Structured Hypermedia: A Matter of Style*
6 Martijn van Welie (VUA) *Task-based User Interface Design*
7 Bastiaan Schonhage (VUA) *Diva: Architectural Perspectives on Information Visualization*
8 Pascal van Eck (VUA) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*
9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models*
10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice*
11 Tom M. van Engers (VUA) *Knowledge Management*

2002

1 Nico Lassing (VUA) *Architecture-Level Modifiability Analysis*
2 Roelof van Zwol (UT) *Modelling and searching web-based document collections*
3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*
4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*
5 Radu Serban (VUA) *The Private Cyberspace Modeling Electronic*
6 Laurens Mommers (UL) *Applied legal epistemology: Building a knowledge-based ontology of*
7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive*
8 Jaap Gordijn (VUA) *Value Based Requirements Engineering: Exploring Innovative*
9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy*
10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*
11 Wouter C. A. Wijngaards (VUA) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*

12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*
13 Hongjing Wu (TUe) *A Reference Architecture for Adaptive Hypermedia Applications*
14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*
15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*
16 Pieter van Langen (VUA) *The Anatomy of Design: Foundations, Models and Applications*
17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VUA) *Ontology-Based Information Sharing in Weakly Structured Environments*
2 Jan Broersen (VUA) *Modal Action Logics for Reasoning About Reactive Systems*
3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*
4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*
5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law: A modelling approach*
6 Boris van Schooten (UT) *Development and specification of virtual environments*
7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*
8 Yongping Ran (UM) *Repair Based Scheduling*
9 Rens Kortmann (UM) *The resolution of visually guided behaviour*
10 Andreas Lincke (UvT) *Electronic Business Negotiation: Some experimental studies on the interaction between medium, innovation context and culture*
11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*
12 Roeland Ordelman (UT) *Dutch speech recognition in multimedia information retrieval*
13 Jeroen Donkers (UM) *Nosce Hostem: Searching with Opponent Models*
14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*
15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*
16 Menzo Windhouwer (CWI) *Feature Grammar Systems: Incremental Maintenance of Indexes to Digital Media Warehouses*
17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*
18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*
2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*
3 Perry Groot (VUA) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*
4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*
5 Viara Popova (EUR) *Knowledge discovery and monotonicity*
6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*
7 Elise Boltjes (UM) *Voorbeeldig onderwijs: voorbeeldgestuurd onderwijs, een opstap naar abstract denken, vooral voor meisjes*
8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale politiële gegevensuitwisseling en digitale expertise*
9 Martin Caminada (VUA) *For the Sake of the Argument: explorations into argument-based reasoning*
10 Suzanne Kabel (UvA) *Knowledge-rich indexing of learning-objects*
11 Michel Klein (VUA) *Change Management for Distributed Ontologies*
12 The Duy Bui (UT) *Creating emotions and facial expressions for embodied agents*
13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*
14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*
15 Arno Knobbe (UU) *Multi-Relational Data Mining*
16 Federico Divina (VUA) *Hybrid Genetic Relational Search for Inductive Learning*
17 Mark Winands (UM) *Informed Search in Complex Games*
18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*
19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*
20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*
2 Erik van der Werf (UM) *AI techniques for the game of Go*
3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*

4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*
5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*
6 Pieter Spronck (UM) *Adaptive Game AI*
7 Flavius Frasincar (TUe) *Hypermedia Presentation Generation for Semantic Web Information Systems*
8 Richard Vdovjak (TUe) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*
9 Jeen Broekstra (VUA) *Storage, Querying and Inferencing for Semantic Web Languages*
10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*
11 Elth Ogston (VUA) *Agent Based Matchmaking and Clustering: A Decentralized Approach to Search*
12 Csaba Boer (EUR) *Distributed Simulation in Industry*
13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*
14 Borys Omelayenko (VUA) *Web-Service configuration on the Semantic Web: Exploring how semantics meets pragmatics*
15 Tibor Bosse (VUA) *Analysis of the Dynamics of Cognitive Processes*
16 Joris Graaumans (UU) *Usability of XML Query Languages*
17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*
18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*
19 Michel van Dartel (UM) *Situated Representation*
20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*
21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TUe) *Foundations of B2B Electronic Contracting*
2 Cristina Chisalita (VUA) *Contextual issues in the design and use of information technology in organizations*
3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*
4 Marta Sabou (VUA) *Building Web Service Ontologies*
5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*
6 Ziv Baida (VUA) *Software-aided Service Bundling: Intelligent Methods & Tools for Graphical Service Modeling*
7 Marko Smiljanic (UT) *XML schema matching: balancing efficiency and effectiveness by means of clustering*
8 Eelco Herder (UT) *Forward, Back and Home Again: Analyzing User Behavior on the Web*
9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*
10 Ronny Siebes (VUA) *Semantic Routing in Peer-to-Peer Systems*
11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*
12 Bert Bongers (VUA) *Interactivation: Towards an e-cology of people, our technological environment, and the arts*
13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*
14 Johan Hoorn (VUA) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*
18 Valentin Zhizhkun (UvA) *Graph transformation for Natural Language Processing*
19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
21 Bas van Gils (RUN) *Aptness on the Web*
22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*
23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
24 Laura Hollink (VUA) *Semantic Annotation for Retrieval of Visual Resources*
25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*
28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
3 Peter Mika (VUA) *Social Networks and the Semantic Web*

4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*
7 Natasa Jovanovic' (UT) *To Whom It May Concern: Addressee Identification in Face-to-Face Meetings*
8 Mark Hoogendoorn (VUA) *Modeling of Change in Multi-Agent Organizations*
9 David Mobach (VUA) *Agent-Based Mediated Service Negotiation*
10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
11 Natalia Stash (TUe) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
13 Rutger Rienks (UT) *Meetings in Smart Environments: Implications of Progressing Technology*
14 Niek Bergboer (UM) *Context-Based Image Analysis*
15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
18 Bart Orriens (UvT) *On the development an management of adaptive business collaborations*
19 David Levy (UM) *Intimate relationships with artificial partners*
20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
23 Peter Barna (TUe) *Specification of Application Logic in Web Information Systems*
24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*
25 Joost Schalken (VUA) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2 Alexei Sharpanskykh (VUA) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
3 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*
4 Ander de Keijzer (UT) *Management of Uncertain Data: towards unattended integration*
5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*
10 Wauter Bosma (UT) *Discourse oriented summarization*
11 Vera Kartseva (VUA) *Designing Controls for Network Organizations: A Value-Based Approach*
12 Jozsef Farkas (RUN) *A Semiotically Oriented Cognitive Model of Knowledge Representation*
13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*
14 Arthur van Bunningen (UT) *Context-Aware Querying: Better Answers with Less Effort*
15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*
16 Henriette van Vugt (VUA) *Embodied agents from a user's perspective*
17 Martin Op 't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*
18 Guido de Croon (UM) *Adaptive Active Vision*
19 Henning Rode (UT) *From Document to Entity Retrieval: Improving Precision and Performance of Focused Text Search*
20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met de overheid op de administratieve lasten van bedrijven*
21 Krisztian Balog (UvA) *People Search in the Enterprise*
22 Henk Koning (UU) *Communication of IT-Architecture*
23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*
24 Zharko Aleksovski (VUA) *Using background knowledge in ontology matching*

25 Geert Jonker (UU) *Efficient and Equitable Exchange in Air Traffic Management Plan Repair using Spender-signed Currency*
26 Marijn Huijbregts (UT) *Segmentation, Diarization and Speech Transcription: Surprise Data Unraveled*
27 Hubert Vogten (OU) *Design and Implementation Strategies for IMS Learning Design*
28 Ildiko Flesch (RUN) *On the Use of Independence Relations in Bayesian Networks*
29 Dennis Reidsma (UT) *Annotations and Subjective Machines: Of Annotators, Embodied Agents, Users, and Other Humans*
30 Wouter van Atteveldt (VUA) *Semantic Network Analysis: Techniques for Extracting, Representing and Querying Media Content*
31 Loes Braun (UM) *Pro-Active Medical Information Retrieval*
32 Trung H. Bui (UT) *Toward Affective Dialogue Management using Partially Observable Markov Decision Processes*
33 Frank Terpstra (UvA) *Scientific Workflow Design: theoretical and practical issues*
34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*
35 Ben Torben Nielsen (UvT) *Dendritic morphologies: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*
2 Willem Robert van Hage (VUA) *Evaluating Ontology-Alignment Techniques*
3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*
4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*
5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks: Based on Knowledge, Cognition, and Quality*
6 Muhammad Subianto (UU) *Understanding Classification*
7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*
8 Volker Nannen (VUA) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*
9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*
10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*
11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*
12 Peter Massuthe (TUE, Humboldt-Universitaet zu Berlin) *Operating Guidelines for Services*
13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*
14 Maksym Korotkiy (VUA) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*
15 Rinke Hoekstra (UvA) *Ontology Representation: Design Patterns and Ontologies that Make Sense*
16 Fritz Reul (UvT) *New Architectures in Computer Chess*
17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*
18 Fabian Groffen (CWI) *Armada, An Evolving Database System*
19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*
20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*
21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*
22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*
23 Peter Hofgesang (VUA) *Modelling Web Usage in a Changing Environment*
24 Annerieke Heuvelink (VUA) *Cognitive Models for Training Simulations*
25 Alex van Ballegooij (CWI) *RAM: Array Database Management through Relational Mapping*
26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*
27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*
28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*
29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*
30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*
31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*
32 Rik Farenhorst (VUA) *Architectural Knowledge Management: Supporting Architects and Auditors*
33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*
34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*
35 Wouter Koelewijn (UL) *Privacy en Politiegegevens: Over geautomatiseerde normatieve informatie-uitwisseling*
36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*
37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*
38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*

39 Christian Stahl (TUE, Humboldt-Universitaet zu Berlin) *Service Substitution: A Behavioral Approach Based on Petri Nets*

40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*

41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*

42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*

43 Virginia Nunes Leal Franqueira (UT) *Finding Multi-step Attacks in Computer Networks using Heuristic Search and Mobile Ambients*

44 Roberto Santana Tapia (UT) *Assessing Business-IT Alignment in Networked Organizations*

45 Jilles Vreeken (UU) *Making Pattern Mining Useful*

46 Loredana Afanasiev (UvA) *Querying XML: Benchmarks and Recursion*

2010

1 Matthijs van Leeuwen (UU) *Patterns that Matter*

2 Ingo Wassink (UT) *Work flows in Life Science*

3 Joost Geurts (CWI) *A Document Engineering Model and Processing Framework for Multimedia documents*

4 Olga Kulyk (UT) *Do You Know What I Know? Situational Awareness of Co-located Teams in Multidisplay Environments*

5 Claudia Hauff (UT) *Predicting the Effectiveness of Queries and Retrieval Systems*

6 Sander Bakkes (UvT) *Rapid Adaptation of Video Game AI*

7 Wim Fikkert (UT) *Gesture interaction at a Distance*

8 Krzysztof Siewicz (UL) *Towards an Improved Regulatory Framework of Free Software. Protecting user freedoms in a world of software communities and eGovernments*

9 Hugo Kielman (UL) *A Politiele gegevensverwerking en Privacy, Naar een effectieve waarborging*

10 Rebecca Ong (UL) *Mobile Communication and Protection of Children*

11 Adriaan Ter Mors (TUD) *The world according to MARP: Multi-Agent Route Planning*

12 Susan van den Braak (UU) *Sensemaking software for crime analysis*

13 Gianluigi Folino (RUN) *High Performance Data Mining using Bio-inspired techniques*

14 Sander van Splunter (VUA) *Automated Web Service Reconfiguration*

15 Lianne Bodenstaff (UT) *Managing Dependency Relations in Inter-Organizational Models*

16 Sicco Verwer (TUD) *Efficient Identification of Timed Automata, theory and practice*

17 Spyros Kotoulas (VUA) *Scalable Discovery of Networked Resources: Algorithms, Infrastructure, Applications*

18 Charlotte Gerritsen (VUA) *Caught in the Act: Investigating Crime by Agent-Based Simulation*

19 Henriette Cramer (UvA) *People's Responses to Autonomous and Adaptive Systems*

20 Ivo Swartjes (UT) *Whose Story Is It Anyway? How Improv Informs Agency and Authorship of Emergent Narrative*

21 Harold van Heerde (UT) *Privacy-aware data management by means of data degradation*

22 Michiel Hildebrand (CWI) *End-user Support for Access to Heterogeneous Linked Data*

23 Bas Steunebrink (UU) *The Logical Structure of Emotions*

24 Zulfiqar Ali Memon (VUA) *Modelling Human-Awareness for Ambient Agents: A Human Mindreading Perspective*

25 Ying Zhang (CWI) *XRPC: Efficient Distributed Query Processing on Heterogeneous XQuery Engines*

26 Marten Voulon (UL) *Automatisch contracteren*

27 Arne Koopman (UU) *Characteristic Relational Patterns*

28 Stratos Idreos (CWI) *Database Cracking: Towards Auto-tuning Database Kernels*

29 Marieke van Erp (UvT) *Accessing Natural History: Discoveries in data cleaning, structuring, and retrieval*

30 Victor de Boer (UvA) *Ontology Enrichment from Heterogeneous Sources on the Web*

31 Marcel Hiel (UvT) *An Adaptive Service Oriented Architecture: Automatically solving Interoperability Problems*

32 Robin Aly (UT) *Modeling Representation Uncertainty in Concept-Based Multimedia Retrieval*

33 Teduh Dirgahayu (UT) *Interaction Design in Service Compositions*

34 Dolf Trieschnigg (UT) *Proof of Concept: Concept-based Biomedical Information Retrieval*

35 Jose Janssen (OU) *Paving the Way for Lifelong Learning: Facilitating competence development through a learning path specification*

36 Niels Lohmann (TUe) *Correctness of services and their composition*

37 Dirk Fahland (TUe) *From Scenarios to components*

38 Ghazanfar Farooq Siddiqui (VUA) *Integrative modeling of emotions in virtual agents*

39 Mark van Assem (VUA) *Converting and Integrating Vocabularies for the Semantic Web*

40 Guillaume Chaslot (UM) *Monte-Carlo Tree Search*

41 Sybren de Kinderen (VUA) *Needs-driven service bundling in a multi-supplier setting: the computational e3-service approach*

42 Peter van Kranenburg (UU) *A Computational Approach to Content-Based Retrieval of Folk Song Melodies*
43 Pieter Bellekens (TUe) *An Approach towards Context-sensitive and User-adapted Access to Heterogeneous Data Sources, Illustrated in the Television Domain*
44 Vasilios Andrikopoulos (UvT) *A theory and model for the evolution of software services*
45 Vincent Pijpers (VUA) *e3alignment: Exploring Inter-Organizational Business-ICT Alignment*
46 Chen Li (UT) *Mining Process Model Variants: Challenges, Techniques, Examples*
47 Jahn-Takeshi Saito (UM) *Solving difficult game positions*
48 Bouke Huurnink (UvA) *Search in Audiovisual Broadcast Archives*
49 Alia Khairia Amin (CWI) *Understanding and supporting information seeking tasks in multiple sources*
50 Peter-Paul van Maanen (VUA) *Adaptive Support for Human-Computer Teams: Exploring the Use of Cognitive Models of Trust and Attention*
51 Edgar Meij (UvA) *Combining Concepts and Language Models for Information Access*

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*
2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*
3 Jan Martijn van der Werf (TUe) *Compositional Design and Verification of Component-Based Information Systems*
4 Hado van Hasselt (UU) *Insights in Reinforcement Learning: Formal analysis and empirical evaluation of temporal-difference*
5 Base van der Raadt (VUA) *Enterprise Architecture Coming of Age: Increasing the Performance of an Emerging Discipline*
6 Yiwen Wang (TUe) *Semantically-Enhanced Recommendations in Cultural Heritage*
7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*
8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*
9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*
10 Bart Bogaert (UvT) *Cloud Content Contention*
11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*
12 Carmen Bratosin (TUe) *Grid Architecture for Distributed Process Mining*
13 Xiaoyu Mao (UvT) *Airport under Control. Multi-agent Scheduling for Airport Ground Handling*
14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*
15 Marijn Koolen (UvA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*
16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*
17 Jiyin He (UvA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*
18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*
19 Ellen Rusman (OU) *The Mind ' s Eye on Personal Profiles*
20 Qing Gu (VUA) *Guiding service-oriented software engineering: A view-based approach*
21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*
22 Junte Zhang (UvA) *System Evaluation of Archival Description and Access*
23 Wouter Weerkamp (UvA) *Finding People and their Utterances in Social Media*
24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*
25 Syed Waqar ul Qounain Jaffry (VUA) *Analysis and Validation of Models for Trust Dynamics*
26 Matthijs Aart Pontier (VUA) *Virtual Agents for Human Communication: Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*
27 Aniel Bhulai (VUA) *Dynamic website optimization through autonomous management of design patterns*
28 Rianne Kaptein (UvA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*
29 Faisal Kamiran (TUe) *Discrimination-aware Classification*
30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*
31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*
32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*
33 Tom van der Weide (UU) *Arguing to Motivate Decisions*
34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*
35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*
36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*

37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*
38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*
39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*
40 Viktor Clerc (VUA) *Architectural Knowledge Management in Global Software Development*
41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*
42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*
43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*
44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*
45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*
46 Beibei Hu (TUD) *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*
47 Azizi Bin Ab Aziz (VUA) *Exploring Computational Models for Intelligent Support of Persons with Depression*
48 Mark Ter Maat (UT) *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*
49 Andreea Niculescu (UT) *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

1 Terry Kakeeto (UvT) *Relationship Marketing for SMEs in Uganda*
2 Muhammad Umair (VUA) *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*
3 Adam Vanya (VUA) *Supporting Architecture Evolution by Mining Software Repositories*
4 Jurriaan Souer (UU) *Development of Content Management System-based Web Applications*
5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*
6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*
7 Rianne van Lambalgen (VUA) *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*
8 Gerben de Vries (UvA) *Kernel Methods for Vessel Trajectories*
9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*
10 David Smits (TUe) *Towards a Generic Distributed Adaptive Hypermedia Environment*
11 J. C. B. Rantham Prabhakara (TUe) *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*
12 Kees van der Sluijs (TUe) *Model Driven Design and Data Integration in Semantic Web Information Systems*
13 Suleman Shahid (UvT) *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*
14 Evgeny Knutov (TUe) *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*
15 Natalie van der Wal (VUA) *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes*
16 Fiemke Both (VUA) *Helping people by understanding them: Ambient Agents supporting task execution and depression treatment*
17 Amal Elgammal (UvT) *Towards a Comprehensive Framework for Business Process Compliance*
18 Eltjo Poort (VUA) *Improving Solution Architecting Practices*
19 Helen Schonenberg (TUe) *What's Next? Operational Support for Business Process Execution*
20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*
21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*
22 Thijs Vis (UvT) *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*
23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*
24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*
25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*
26 Emile de Maat (UvA) *Making Sense of Legal Text*
27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*
28 Nancy Pascall (UvT) *Engendering Technology Empowering Women*
29 Almer Tigelaar (UT) *Peer-to-Peer Information Retrieval*
30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*
31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*
32 Wietske Visser (TUD) *Qualitative multi-criteria preference representation and reasoning*

33 Rory Sie (OUN) *Coalitions in Cooperation Networks (COCOON)*
34 Pavol Jancura (RUN) *Evolutionary analysis in PPI networks and applications*
35 Evert Haasdijk (VUA) *Never Too Old To Learn: On-line Evolution of Controllers in Swarm- and Modular Robotics*
36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*
37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*
38 Selmar Smit (VUA) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*
39 Hassan Fatemi (UT) *Risk-aware design of value and coordination networks*
40 Agus Gunawan (UvT) *Information Access for SMEs in Indonesia*
41 Sebastian Kelle (OU) *Game Design Patterns for Learning*
42 Dominique Verpoorten (OU) *Reflection Amplifiers in self-regulated Learning*
43 Anna Tordai (VUA) *On Combining Alignment Techniques*
44 Benedikt Kratz (UvT) *A Model and Language for Business-aware Transactions*
45 Simon Carter (UvA) *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*
46 Manos Tsagkias (UvA) *Mining Social Media: Tracking Content and Predicting Behavior*
47 Jorn Bakker (TUe) *Handling Abrupt Changes in Evolving Time-series Data*
48 Michael Kaisers (UM) *Learning against Learning: Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*
49 Steven van Kervel (TUD) *Ontologogy driven Enterprise Information Systems Engineering*
50 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling: a practical framework with a case study in elevator dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*
2 Erietta Liarou (CWI) *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*
3 Szymon Klarman (VUA) *Reasoning with Contexts in Description Logics*
4 Chetan Yadati (TUD) *Coordinating autonomous planning and scheduling*
5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*
6 Romulo Goncalves (CWI) *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*
7 Giel van Lankveld (UvT) *Quantifying Individual Player Differences*
8 Robbert-Jan Merk (VUA) *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*
9 Fabio Gori (RUN) *Metagenomic Data Analysis: Computational Methods and Applications*
10 Jeewanie Jayasinghe Arachchige (UvT) *A Unified Modeling Framework for Service Design*
11 Evangelos Pournaras (TUD) *Multi-level Reconfigurable Self-organization in Overlay Services*
12 Marian Razavian (VUA) *Knowledge-driven Migration to Services*
13 Mohammad Safiri (UT) *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*
14 Jafar Tanha (UvA) *Ensemble Approaches to Semi-Supervised Learning Learning*
15 Daniel Hennes (UM) *Multiagent Learning: Dynamic Games and Applications*
16 Eric Kok (UU) *Exploring the practical benefits of argumentation in multi-agent deliberation*
17 Koen Kok (VUA) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*
18 Jeroen Janssens (UvT) *Outlier Selection and One-Class Classification*
19 Renze Steenhuizen (TUD) *Coordinated Multi-Agent Planning and Scheduling*
20 Katja Hofmann (UvA) *Fast and Reliable Online Learning to Rank for Information Retrieval*
21 Sander Wubben (UvT) *Text-to-text generation by monolingual machine translation*
22 Tom Claassen (RUN) *Causal Discovery and Logic*
23 Patricio de Alencar Silva (UvT) *Value Activity Monitoring*
24 Haitham Bou Ammar (UM) *Automated Transfer in Reinforcement Learning*
25 Agnieszka Anna Latoszek-Berendsen (UM) *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*
26 Alireza Zarghami (UT) *Architectural Support for Dynamic Homecare Service Provisioning*
27 Mohammad Huq (UT) *Inference-based Framework Managing Data Provenance*
28 Frans van der Sluis (UT) *When Complexity becomes Interesting: An Inquiry into the Information eXperience*
29 Iwan de Kok (UT) *Listening Heads*
30 Joyce Nakatumba (TUe) *Resource-Aware Business Process Management: Analysis and Support*
31 Dinh Khoa Nguyen (UvT) *Blueprint Model and Language for Engineering Cloud Applications*
32 Kamakshi Rajagopal (OUN) *Networking For Learning: The role of Networking in a Lifelong Learner's Professional Development*

33 Qi Gao (TUD) *User Modeling and Personalization in the Microblogging Sphere*
34 Kien Tjin-Kam-Jet (UT) *Distributed Deep Web Search*
35 Abdallah El Ali (UvA) *Minimal Mobile Human Computer Interaction*
36 Than Lam Hoang (TUe) *Pattern Mining in Data Streams*
37 Dirk Börner (OUN) *Ambient Learning Displays*
38 Eelco den Heijer (VUA) *Autonomous Evolutionary Art*
39 Joop de Jong (TUD) *A Method for Enterprise Ontology based Design of Enterprise Information Systems*
40 Pim Nijssen (UM) *Monte-Carlo Tree Search for Multi-Player Games*
41 Jochem Liem (UvA) *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*
42 Léon Planken (TUD) *Algorithms for Simple Temporal Reasoning*
43 Marc Bron (UvA) *Exploration and Contextualization through Interaction and Concepts*

2014

1 Nicola Barile (UU) *Studies in Learning Monotone Models from Data*
2 Fiona Tuliyano (RUN) *Combining System Dynamics with a Domain Modeling Method*
3 Sergio Raul Duarte Torres (UT) *Information Retrieval for Children: Search Behavior and Solutions*
4 Hanna Jochmann-Mannak (UT) *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*
5 Jurriaan van Reijsen (UU) *Knowledge Perspectives on Advancing Dynamic Capability*
6 Damian Tamburri (VUA) *Supporting Networked Software Development*
7 Arya Adriansyah (TUe) *Aligning Observed and Modeled Behavior*
8 Samur Araujo (TUD) *Data Integration over Distributed and Heterogeneous Data Endpoints*
9 Philip Jackson (UvT) *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*
10 Ivan Salvador Razo Zapata (VUA) *Service Value Networks*
11 Janneke van der Zwaan (TUD) *An Empathic Virtual Buddy for Social Support*
12 Willem van Willigen (VUA) *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*
13 Arlette van Wissen (VUA) *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*
14 Yangyang Shi (TUD) *Language Models With Meta-information*
15 Natalya Mogles (VUA) *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*
16 Krystyna Milian (VUA) *Supporting trial recruitment and design by automatically interpreting eligibility criteria*
17 Kathrin Dentler (VUA) *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*
18 Mattijs Ghijsen (UvA) *Methods and Models for the Design and Study of Dynamic Agent Organizations*
19 Vinicius Ramos (TUe) *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*
20 Mena Habib (UT) *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*
21 Kassidy Clark (TUD) *Negotiation and Monitoring in Open Environments*
22 Marieke Peeters (UU) *Personalized Educational Games: Developing agent-supported scenario-based training*
23 Eleftherios Sidirourgos (UvA/CWI) *Space Efficient Indexes for the Big Data Era*
24 Davide Ceolin (VUA) *Trusting Semi-structured Web Data*
25 Martijn Lappenschaar (RUN) *New network models for the analysis of disease interaction*
26 Tim Baarslag (TUD) *What to Bid and When to Stop*
27 Rui Jorge Almeida (EUR) *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*
28 Anna Chmielowiec (VUA) *Decentralized k-Clique Matching*
29 Jaap Kabbedijk (UU) *Variability in Multi-Tenant Enterprise Software*
30 Peter de Cock (UvT) *Anticipating Criminal Behaviour*
31 Leo van Moergestel (UU) *Agent Technology in Agile Multiparallel Manufacturing and Product Support*
32 Naser Ayat (UvA) *On Entity Resolution in Probabilistic Data*
33 Tesfa Tegegne (RUN) *Service Discovery in eHealth*
34 Christina Manteli (VUA) *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems*
35 Joost van Ooijen (UU) *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*
36 Joos Buijs (TUe) *Flexible Evolutionary Algorithms for Mining Structured Process Models*

37 Maral Dadvar (UT) *Experts and Machines United Against Cyberbullying*
38 Danny Plass-Oude Bos (UT) *Making brain-computer interfaces better: improving usability through post-processing*
39 Jasmina Maric (UvT) *Web Communities, Immigration, and Social Capital*
40 Walter Omona (RUN) *A Framework for Knowledge Management Using ICT in Higher Education*
41 Frederic Hogenboom (EUR) *Automated Detection of Financial Events in News Text*
42 Carsten Eijckhof (CWI/TUD) *Contextual Multidimensional Relevance Models*
43 Kevin Vlaanderen (UU) *Supporting Process Improvement using Method Increments*
44 Paulien Meesters (UvT) *Intelligent Blauw: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden*
45 Birgit Schmitz (OUN) *Mobile Games for Learning: A Pattern-Based Approach*
46 Ke Tao (TUD) *Social Web Data Analytics: Relevance, Redundancy, Diversity*
47 Shangsong Liang (UvA) *Fusion and Diversification in Information Retrieval*

2015

1 Niels Netten (UvA) *Machine Learning for Relevance of Information in Crisis Response*
2 Faiza Bukhsh (UvT) *Smart auditing: Innovative Compliance Checking in Customs Controls*
3 Twan van Laarhoven (RUN) *Machine learning for network data*
4 Howard Spoelstra (OUN) *Collaborations in Open Learning Environments*
5 Christoph Bösch (UT) *Cryptographically Enforced Search Pattern Hiding*
6 Farideh Heidari (TUD) *Business Process Quality Computation: Computing Non-Functional Requirements to Improve Business Processes*
7 Maria-Hendrike Peetz (UvA) *Time-Aware Online Reputation Analysis*
8 Jie Jiang (TUD) *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*
9 Randy Klaassen (UT) *HCI Perspectives on Behavior Change Support Systems*
10 Henry Hermans (OUN) *OpenU: design of an integrated system to support lifelong learning*
11 Yongming Luo (TUe) *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*
12 Julie M. Birkholz (VUA) *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*
13 Giuseppe Procaccianti (VUA) *Energy-Efficient Software*
14 Bart van Straalen (UT) *A cognitive approach to modeling bad news conversations*
15 Klaas Andries de Graaf (VUA) *Ontology-based Software Architecture Documentation*
16 Changyun Wei (UT) *Cognitive Coordination for Cooperative Multi-Robot Teamwork*
17 André van Cleeff (UT) *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*
18 Holger Pirk (CWI) *Waste Not, Want Not!: Managing Relational Data in Asymmetric Memories*
19 Bernardo Tabuenca (OUN) *Ubiquitous Technology for Lifelong Learners*
20 Loïs Vanhée (UU) *Using Culture and Values to Support Flexible Coordination*
21 Sibren Fetter (OUN) *Using Peer-Support to Expand and Stabilize Online Learning*
22 Zhemin Zhu (UT) *Co-occurrence Rate Networks*
23 Luit Gazendam (VUA) *Cataloguer Support in Cultural Heritage*
24 Richard Berendsen (UvA) *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*
25 Steven Woudenberg (UU) *Bayesian Tools for Early Disease Detection*
26 Alexander Hogenboom (EUR) *Sentiment Analysis of Text Guided by Semantics and Structure*
27 Sándor Héman (CWI) *Updating compressed colomn stores*
28 Janet Bagorogoza (TiU) *KNOWLEDGE MANAGEMENT AND HIGH PERFORMANCE: The Uganda Financial Institutions Model for HPO*
29 Hendrik Baier (UM) *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*
30 Kiavash Bahreini (OU) *Real-time Multimodal Emotion Recognition in E-Learning*
31 Yakup Koç (TUD) *On the robustness of Power Grids*
32 Jerome Gard (UL) *Corporate Venture Management in SMEs*
33 Frederik Schadd (TUD) *Ontology Mapping with Auxiliary Resources*
34 Victor de Graaf (UT) *Gesocial Recommender Systems*
35 Jungxao Xu (TUD) *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

2016

1 Syed Saiden Abbas (RUN) *Recognition of Shapes by Humans and Machines*
2 Michiel Christiaan Meulendijk (UU) *Optimizing medication reviews through decision support: prescribing a better pill to swallow*

3 Maya Sappelli (RUN) *Knowledge Work in Context: User Centered Knowledge Worker Support*
4 Laurens Rietveld (VU) *Publishing and Consuming Linked Data*
5 Evgeny Sherkhonov (UvA) *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*
6 Michel Wilson (TUD) *Robust scheduling in an uncertain environment*
7 Jeroen de Man (VU) *Measuring and modeling negative emotions for virtual training*
8 Matje van de Camp (TiU) *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*
9 Archana Nottamkandath (VU) *Trusting Crowdsourced Information on Cultural Artefacts*
10 George Karafotias (VUA) *Parameter Control for Evolutionary Algorithms*
11 Anne Schuth (UvA) *Search Engines that Learn from Their Users*
12 Max Knobbout (UU) *Logics for Modelling and Verifying Normative Multi-Agent Systems*
13 Nana Baah Gyan (VU) *The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach*
14 Ravi Khadka (UU) *Revisiting Legacy Software System Modernization*
15 Steffen Michels (RUN) *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments*
16 Guangliang Li (UvA) *Socially Intelligent Autonomous Agents that Learn from Human Reward*
17 Berend Weel (VU) *Towards Embodied Evolution of Robot Organisms*
18 Albert Meroño Peñuela (VU) *Refining Statistical Data on the Web*
19 Julia Efremova (Tu/e) *Mining Social Structures from Genealogical Data*
20 Daan Odijk (UvA) *Context & Semantics in News & Web Search*
21 Alejandro Moreno Clleri (UT) *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*
22 Grace Lewis (VU) *Software Architecture Strategies for Cyber-Foraging Systems*
23 Fei Cai (UvA) *Query Auto Completion in Information Retrieval*