
Modeling User Information Needs on Mobile Devices

From Recommendation to Conversation

Doctoral Dissertation submitted to the
Faculty of Informatics of the Università della Svizzera Italiana
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

presented by
Mohammad Aliannejadi

under the supervision of
Fabio Crestani

September 2019

Dissertation Committee

Fabio Crestani	Università della Svizzera italiana (USI), Lugano, Switzerland
Josiane Mothe	Université Toulouse - Jean Jaurès, Toulouse, France
Stefano Mizzaro	Università degli Studi di Udine, Udine, Italy
Laura Pozzi	Università della Svizzera italiana (USI), Lugano, Switzerland
Antonio Carzaniga	Università della Svizzera italiana (USI), Lugano, Switzerland

Dissertation accepted on 13 September 2019

Research Advisor

Fabio Crestani

PhD Program Director

Walter Binder and Silvia Santini

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Mohammad Aliannejadi
Lugano, 13 September 2019

In memory of my father

“If we knew what it was we were
doing, it would not be called
research, would it?”

Albert Einstein

Abstract

Recent advances in the development of mobile devices, equipped with multiple sensors, together with the availability of millions of applications have made these devices more pervasive in our lives than ever. The availability of the diverse set of sensors, as well as high computational power, enable information retrieval (IR) systems to sense a user’s context and personalize their results accordingly. Relevant studies show that people use their mobile devices to access information in a wide range of topics in various contextual situations, highlighting the fact that modeling user information need on mobile devices involves studying several means of information access.

In this thesis, we study three major aspects of information access on mobile devices. First, we focus on proactive approaches to *modeling users for venue suggestion*. We investigate three methods of user modeling, namely, content-based, collaborative, and hybrid, focusing on personalization and context-awareness. We propose a two-phase collaborative ranking algorithm for leveraging users’ implicit feedback while incorporating temporal and geographical information into the model. We then extend our collaborative model to include multiple cross-venue similarity scores and combine it with our content-based approach to produce a hybrid recommendation.

Second, we introduce and investigate a new task on mobile search, that is, *unified mobile search*. We take the first step in defining, studying, and modeling this task by collecting two datasets and conducting experiments on one of the main components of unified mobile search frameworks, that is target apps selection. To this end, we propose two neural approaches.

Finally, we address the *conversational aspect of mobile search* where we propose an offline evaluation protocol and build a dataset for asking clarifying questions for conversational search. Also, we propose a retrieval framework consisting of three main components: question retrieval, question selection, and document retrieval. The experiments and analyses indicate that asking clarifying questions should be an essential part of a conversational system, resulting in high performance gain.

Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Fabio Crestani for the continuous support of my Ph.D. study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis.

Besides my advisor, I would like to thank the rest of my thesis committee: Prof. Josiane Mothe, Prof. Laura Pozzi, Prof. Stefano Mizzaro, and Prof. Antonio Carzaniga for their insightful comments and encouragement.

My sincere thanks also go to Prof. W. Bruce Croft, who hosted me at the Center for Intelligent Information Retrieval (CIIR) at the University of Massachusetts Amherst, to join his group as a visiting scholar. Without his precious support and contributions, it would not be possible to conduct this research. During and before my visit, I had the opportunity to work with Dr. Hamed Zamani, to whom I am grateful for the stimulating discussions and incredible collaboration we have had in the past two years. I would also thank other members of the CIIR, namely, Hamed Bonab, and Helia Hashemi, with whom I had the pleasure of working.

I would also like to thank my collaborators, who helped me throughout my Ph.D. via countless meetings that we had. In particular, I am grateful to Dr. Dimitrios Rafailidis, Dr. Morgan Harvey, and Matthew Pointon. I also thank my colleagues for the constructive discussions, for the sleepless nights we were working together before deadlines, and for all the fun we have had in the past few years. In particular, I would like to thank Dr. Ali Bahrainian, Dr. Ida Mele, Luca Costa, Dr. Monica Landoni, Maram Barifah, Dr. Anastasia Giachanou, Esteban Andrés Ríssola, and Manajit Chakraborty. Also, I thank my friends at Università della Svizzera italiana, as well as my Italian language teacher, Pamela Trincado, who patiently helped me when it was difficult to catch up with the class due to the heavy workload.

Last but not least, I would like to express my deepest gratitude and appreciation to my mother, my wife, and the rest of my family for their unconditional love and support throughout my Ph.D., writing this thesis, and my life.

Contents

Contents	xi
List of List of Figures	xvii
List of List of Tables	xxi
1 Introduction	1
1.1 Motivation	1
1.2 Thesis Outline	4
1.3 Main Contributions	5
1.4 Resources Created and Released	7
1.5 Publication Overview	8
1.6 Additional Publications	10
2 Literature Review	13
2.1 Venue Suggestion	13
2.1.1 TREC Contextual Suggestion	15
2.1.2 Context-Aware POI Recommendation	16
2.1.3 Collaborative Ranking	17
2.1.4 Time-Aware Recommendation	18
2.2 Mobile Search	18
2.2.1 Mobile IR	19
2.2.2 Mobile HCI	20
2.2.3 Context-Aware Search	21
2.2.4 Proactive IR	21
2.2.5 Federated and Aggregated Search	22
2.2.6 Query Classification	22
2.3 Conversational Search	23
2.3.1 Conversational IR	23
2.3.2 Clarifying Questions	24

2.3.3	Conversational Question Answering	24
I	Venue Suggestion	27
3	Content-based User Modeling for Venue Suggestion	29
3.1	Introduction	29
3.2	Personalized Keyword Boosting	32
3.2.1	Personalized Keyword-Tag Mapping	32
3.2.2	Parameter Estimation Based on Expectation-Maximization	34
3.2.3	Location Keywords Boosting	35
3.2.4	User Tag Prediction	37
3.3	Contextual Appropriateness Prediction	39
3.3.1	Contextual Features	40
3.3.2	Training the Classifier	41
3.4	Recommendation based on Information from Multiple LBSNs . . .	42
3.4.1	Frequency-based Score	42
3.4.2	Review-Based Score	43
3.4.3	Location Ranking	45
3.5	Data Collection and Analysis	45
3.5.1	Data Crawling	46
3.5.2	Crowdsourcing	47
3.5.3	Data Analysis	48
3.6	Experimental Setup	49
3.6.1	Data	49
3.6.2	Metrics	51
3.6.3	Compared Methods	53
3.7	Results and Discussion	55
3.7.1	Performance Comparison	55
3.7.2	Impact of Different Learning to Rank Techniques	56
3.7.3	Impact of Using Information from Multiple LBSNs	56
3.7.4	Impact of Using Different Scores	59
3.7.5	Impact of Number of Visited POIs	61
3.7.6	Impact of Visiting POIs from a Single City vs. Two Cities .	62
3.7.7	Dimensionality Reduction	63
3.7.8	User Tag Prediction	64
3.8	Summary	65

4 Collaborative User Modeling for Venue Suggestion	67
4.1 Introduction	67
4.2 Data Analysis	69
4.2.1 Data	69
4.2.2 Time-Dependency of User Activities and Interests	69
4.2.3 Users' Multiple Check-ins	71
4.2.4 Remarks	72
4.3 Proposed Method	73
4.3.1 Geographical Similarity	75
4.3.2 Phase 1: Visited vs. Unvisited POIs	75
4.3.3 Phase 2: Multiple vs. Single Check-ins	77
4.3.4 Time-Sensitive Regularizer	78
4.3.5 Joint Two-Phase Collaborative Ranking Algorithm	79
4.4 Experimental Setup	80
4.4.1 Data	81
4.4.2 Metrics	81
4.4.3 Compared Methods	82
4.5 Results and Discussion	83
4.5.1 Performance Comparison	83
4.5.2 Impact of the 2 nd Phase	88
4.5.3 Impact of the Time-Sensitive Regularizer	89
4.5.4 Impact of the Geographical Influence	89
4.5.5 Impact of the Model Parameters	90
4.5.6 Model's Convergence	90
4.6 Summary	91
5 Hybrid User Modeling for Venue Suggestion	95
5.1 Introduction	95
5.2 Proposed Method	97
5.2.1 Collaborative Ranking with Multiple Location-based Similarities	97
5.2.2 Cross-Venue Similarities	99
5.2.3 System Overview	101
5.2.4 Hybrid Venue Suggestion	101
5.3 Experimental Setup	103
5.3.1 Data	103
5.3.2 Metrics	103
5.3.3 Compared Methods	104
5.4 Results and Discussion	105

5.4.1	Performance Comparison	105
5.4.2	Impact of the Number of Visited POIs	106
5.4.3	Impact of the Similarity Scores	108
5.4.4	Impact of the Number of Latent Factors	109
5.4.5	Impact of Regularization Parameter	109
5.5	Summary	109
II Mobile Search		111
6	Unified Mobile Search	113
6.1	Introduction	113
6.2	Data Collection	115
6.3	Data Analysis	118
6.3.1	App Distribution	118
6.3.2	Query Attributes	123
6.3.3	Query Overlap	124
6.3.4	Remarks	125
6.4	Neural Target Apps Selection	125
6.4.1	NTAS1: App Scoring Model	126
6.4.2	NTAS2: Query Classification Model	128
6.5	Experimental Setup	128
6.5.1	Data	128
6.5.2	Metrics	128
6.5.3	Compared Methods	129
6.6	Results and Discussion	130
6.6.1	Performance Comparison	130
6.6.2	Representation Analysis	132
6.6.3	Performance on Apps	132
6.6.4	Performance on Tasks	133
6.7	Summary	135
7	Context-Aware Target Apps Selection	137
7.1	Introduction	137
7.2	Data Collection	139
7.2.1	uSearch	139
7.2.2	Data Collection Procedure	140
7.2.3	Quality Check	141
7.2.4	Privacy Concerns	141

7.2.5	Limitations	142
7.3	Data Analysis	142
7.3.1	Basic Statistics	142
7.3.2	Apps	143
7.3.3	Queries	145
7.3.4	Sessions	147
7.3.5	Context	148
7.4	Context-Aware Neural Target Apps Selection	148
7.5	Experimental Setup	152
7.5.1	Data	152
7.5.2	Metrics	152
7.5.3	Compared Methods	153
7.6	Results and Discussion	154
7.6.1	Performance Comparison	154
7.6.2	Impact of Context on Performance Per App	157
7.6.3	Impact of Context on Performance Per User	157
7.6.4	Impact of Context on Performance Per Query Length.	158
7.7	Summary	158

III Conversational Search 161

8	Conversational Search with Clarifying Questions	163
8.1	Introduction	163
8.2	Problem Statement	166
8.3	Data Collection	168
8.3.1	Topics and Facets	168
8.3.2	Clarifying Questions	170
8.3.3	Question Verification and Addition	171
8.3.4	Answers	171
8.4	Conversational Retrieval Framework	174
8.4.1	Question Retrieval Model	175
8.4.2	Question Selection Model	176
8.4.3	Document Retrieval Model	177
8.5	Experimental Setup	178
8.5.1	Data	178
8.5.2	Metrics	178
8.6	Results and Discussion	181
8.6.1	Question Retrieval	181

8.6.2	Oracle Question Selection	181
8.6.3	Question Selection	182
8.6.4	Impact of Data Splits	183
8.6.5	Impact of Number of Conversation Turns	183
8.6.6	Impact of Clarifying Questions on Facets	183
8.6.7	Case Study: Failure and Success Analysis	187
8.7	Limitations	188
8.8	Summary	189
9	Conclusions	191
9.1	Summary of the Work Carried Out	191
9.2	Main Contributions	192
9.3	Future Research Directions	195
	Bibliography	199

List of Figures

1.1	Mobile users search for a wide variety of information [89].	2
1.2	Mobile search contexts vary by type of search [89].	3
3.1	Overview of the proposed method.	31
3.2	An example of mapping of $J = 4$ location keywords to $I = 2$ user tags.	34
3.3	A sample of tags from three different users assigned to one single location and the calculated mapping. The lines connect each user tag to their mapped location keywords. Also, the index number of the mapped location keywords is written in parentheses for more convenient reading.	36
3.4	Histogram of venue-context appropriateness score ranges. We partition the histogram into 3 parts based on the scores range. Scores below -0.4 represent <i>inappropriateness</i> and score higher than $+0.4$ represent <i>appropriateness</i> . Scores between -0.4 and $+0.4$ do not provide much information and show no agreement among assessors (subjective task).	50
3.5	Effect on P@5 by varying the number of locations that each user has visited for (a) TREC-CS 2015 and (b) TREC-CS 2016.	62
3.6	Our model's performance in terms of P@5 with different number of locations as users' history of preferences compared to LinearCatRev. We have chose the order of locations in two different manners. <i>Sequential</i> : the first 30 locations are from one single city, the second 30 are from another city, <i>Interleaved</i> : the list of locations is interleaved based on their cities.	63
4.1	Number of check-ins per month on Foursquare's and Gowalla's.	71
4.2	Popularity of the top-8 categories over time on Gowalla's (best viewed in color).	72

4.3	Check-in distribution of users and POI categories over time in Gowalla's. Figures a & c depict the least time-variant users and POI categories, respectively. Figures b & d, in contrast, show the distribution of the most time-variant users and POI categories, respectively (best viewed in color).	73
4.4	Check-in histogram of 100 randomly-sampled users from the 500 most active users of Gowalla's. For each user, the red bar denotes the number of multiple check-ins, while the blue bar denotes the number of single check-ins.	74
4.5	Impact of the number of latent factors.	91
4.6	Impact of λ	92
4.7	Impact of α	93
4.8	Convergence of the joint objective function.	93
5.1	Impact of different model parameters on the performance of CR-MLS109	
6.1	Workflow of an example unified mobile search framework.	114
6.2	HIT interface for choosing apps. The workers could enter an app's name or click on an app's icon.	117
6.3	The distribution of number of queries with respect to apps and users.	120
6.4	Number of queries per app for the top 17 apps.	121
6.5	Distribution of unique apps per user and task.	121
6.6	Histogram of number of query terms per app.	122
6.7	Query length distribution with respect to number of terms and characters.	122
6.8	Distribution of top query unigrams for two sample apps.	124
6.9	Proximity of different app representations learned by NTAS1-pairwise. This plot is produced by reducing the dimensionality (using the t-SNE algorithm) of the app representations to two for visualization.	133
6.10	Performance comparison with respect to certain apps on both data splits.	134
6.11	Negative correlation between the number of unique apps users selected for a task and performance.	135
7.1	uSearch interface on LG Google Nexus 5 as well as the survey. Checkboxes are used to indicate the target app for a query.	140
7.2	Number of queries and active participants per day, during the course of data collection (best viewed in color).	144
7.3	Number of queries per app for top 20 apps.	144
7.4	Distribution of unique apps per user and task.	145

7.5	Time-of-the-day distribution of queries and unique apps (best viewed in color).	149
7.6	Apps usage context ranking distribution of relevant target apps. Lower values of x axis mean that the app has been used more often in the past 24 hours.	149
7.7	Performance comparison with respect to certain apps with and without context.	156
7.8	MRR differences on ISTAS-R with and without context per app and user.	156
8.1	Example conversations with clarifying questions from our dataset, Qulac. As we see, both users, Alice and Robin, issue the same query (“dinosaur”), however, their actual information needs are completely different. With no prior knowledge, the system starts with the same clarifying question. Depending on the user’s answers, the system selects the next questions in order to clarify the user’s information need. The tag “No answer” shows that the asked question is not related to the information need. We asked the crowdworkers to answer each question given the original query and information need. In cases where the question required knowledge that was out of the scope of the information need, the workers would mark their answer with a “No answer” tag (see example).	164
8.2	A workflow for asking clarifying questions in an open-domain conversational search system.	166
8.3	An example of three facets with their corresponding relevant documents for the topic “dinosaur” (best viewed in color).	169
8.4	An example of three users who have issued the same query “dinosaur,” but with different information needs. As we see, the faceted relevance assessments are broken into three different sets creating three new topics (best viewed in color).	170
8.5	Screenshots of clarifying question generation HIT instructions. . .	172
8.6	Screenshot of answer generation HIT instructions.	173
8.7	Impact of topic type, facet type, and query length on the performance of BestQuestion oracle model, compared to OriginalQuery.	182
8.8	Performance comparison with the baselines for different number of conversation turns ($k \in \{1, 2, 3\}$).	185

List of Tables

3.1	Description of different contextual information dimensions.	39
3.2	Examples of contextual features generated using crowdsourcing. . .	39
3.3	Four proposed models using different combination of similarity scores.	45
3.4	Statistics on the crawled collection	49
3.5	Statistics on the crowdsourced contextual appropriateness collection	49
3.6	Statistical details of user tagging dataset	51
3.7	Performance evaluation on TREC-CS 2015.	57
3.8	Performance evaluation on TREC-CS 2016.	57
3.9	Effect on P@5 for different learning to rank techniques in TREC-CS 2015.	58
3.10	Effect on P@5 for different learning to rank techniques in TREC-CS 2016.	58
3.11	Performance evaluation after removing information provided by Foursquare (F) and Yelp (Y) in the TREC-CS 2015 dataset.	59
3.12	Performance evaluation after removing information provided by Foursquare (F) and Yelp (Y) in the TREC-CS 2016 dataset.	60
3.13	Performance of PK-Boosting using all the scores (<i>All</i>) and after removing each score at a time.	61
3.14	Performance comparison on TREC-CS 2015 on dimensionality reduction.	65
3.15	Performance comparison on TREC-CS 2016 on dimensionality reduction.	65
3.16	Performance comparison of user tag prediction models.	65
4.1	General statistics of the datasets	70
4.2	Performance evaluation on Foursquare’s in terms of nDCG@k. . .	84
4.3	Performance evaluation on Foursquare’s in terms of P@k.	85
4.4	Performance evaluation on Gowalla’s in terms of nDCG@k.	86

4.5	Performance evaluation on Gowalla’s in terms of P@k.	87
5.1	Performance evaluation on TREC-CS in terms of P@k with $k \in \{1, 2, 3, 4, 5\}$. Bold values denote the best scores compared with collaborative approaches and the content-based approach separately.	107
5.2	Performance evaluation on TREC-CS in terms of nDCG@k with $k \in \{1, 2, 3, 4, 5\}$. Bold values denote the best scores compared with collaborative approaches and the content-based approach separately.	107
5.3	Effect on P@5 and nDCG@5 of different number of venues that users visited as training set.	108
6.1	Distribution of crowdsourcing search task categories.	116
6.2	Statistics of UniMobile.	118
6.3	The percentage of similar queries at different similarity thresholds considering only the queries associated with every app.	126
6.4	Performance comparison with baselines on UniMobile-Q and UniMobile-T.	131
7.1	Statistics of ISTAS.	143
7.2	Corss-app query attributes for 9 apps.	146
7.3	Performance comparison with baselines on ISTAS-R and ISTAS-T.	155
7.4	Performance analysis based on query length, dividing the test queries into three evenly-sized length buckets.	157
8.1	Statistics of Qulac.	174
8.2	Performance of question retrieval model.	181
8.3	Performance comparison with baselines on Qulac-T and Qulac-F.	184
8.4	Failure and success examples of NeuQS. Failure and success are measured by the difference in performance of NeuQS and OriginalQuery in terms of MRR (Δ MRR).	186

Chapter 1

Introduction

1.1 Motivation

Recent years have witnessed a rapid growth in the use of mobile devices, such as smartphones and tablets, to search the web for information. This has resulted in a shift of users' behavior to the extent that, as of 2016, more searches are performed on mobile devices than on more "traditional" desktop computers [87]. Mobile devices can be readily used in many situations that a desktop computer cannot since they are carried constantly on one's person at most of the times of the day and night. As such, searching is now performed in a larger range of contexts than ever before and, often, at the same time with other tasks [94].

Moreover, mobile devices are nowadays equipped with many sensors such as GPS, light sensor, and accelerometer. These sensors enable a system to capture a user's current position, surrounding devices, movement, people they are with, time of the day, and weather as their current context. A mobile IR system should be able to model a user's context and interest to personalize the retrieval process according to the user's needs and context. Therefore, search on mobile devices has become an area of interest in Information Retrieval (IR) [73].

Users access information on their mobile devices to address a wide range of information needs such as shopping, dining, or visiting a new place. In particular, Google and Nielsen conducted a large-scale study in 2013 to understand how people access information on their mobile devices [89]. As we see in Figure 1.1, they found that mobile users search for a wide variety of information [89] with Art & Entertainment and News being the top search topics. Users tend to access a wide range of information in various circumstances such as at home, on the go, and in-store. We see in Figure 1.2 that Google and Nielsen [89] also found mobile search contexts vary by type of search. Specifically, we see that many

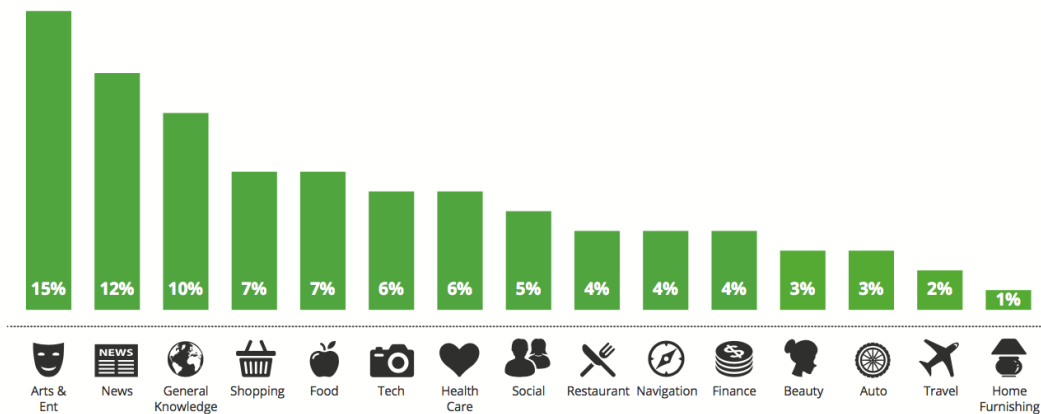


Figure 1.1. Mobile users search for a wide variety of information [89].

types of mobile searches occur out of the home. For instance, people search for food and shopping, mainly while they are in a store or on the go.

Using mobile devices to access various types of information under various contexts requires multiple strategies for modeling users' information needs. Modeling information needs of mobile users should consider various aspects that are related to mobile information access. In particular, it should take into account multiple methods of user interaction such as voice, text, and gestures under various contexts. For instance, suppose a user who is visiting a new city and is walking in the city center. In this context, the user's attention is fragmented [94] since they have to pay attention to their surrounding area while walking and, as a consequence, their interaction with their device is much limited. Therefore, the system should be able to anticipate the user's information need and recommend points of interest (POIs) to them. Alternatively, the system can provide the user with a voice interface through which the user can communicate their information need with their device more conveniently.

Considering both the type of information need and context, we have focused on three main aspects of mobile information access:

1. A mobile information access system must be able to model the user information need according to their past behavior, as well as their current context, to provide personalized context-aware recommendations.
2. Such a system must be able to understand the users' queries under different contexts to retrieve relevant information from multiple applications. In many cases, different parts of information need should be retrieved from different applications and put together.

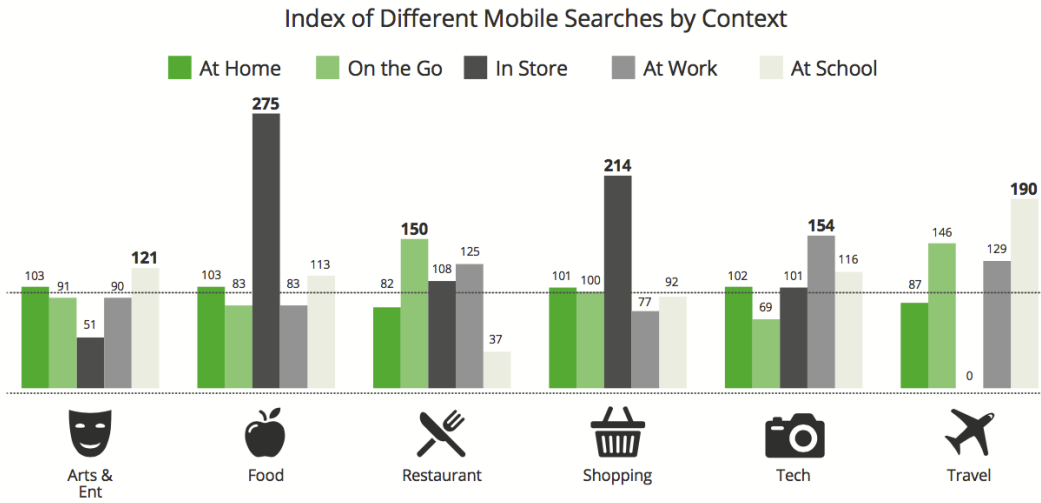


Figure 1.2. Mobile search contexts vary by type of search [89].

3. The system must converse with the user to understand their information need. Many queries are incomplete, faceted, or ambiguous and, as such, a conversational search system needs to ask clarifying questions whose answers would help the system understand the user's information needs more clearly.

These three aspects constitute the main ideas behind the new generation of personal assistants such as Google Now, Microsoft Cortana, and Apple Siri. The main goal of these systems is to be able to converse with users effectively and retrieve useful information from multiple channels such as applications and websites. At the same time, they aim to provide helpful information to the users based on their past behavior and current context.

In this thesis, we present our work on all these three aspects of mobile information access. First, we study the problem of venue suggestion for mobile devices as it enables proactive information retrieval on mobile devices. Also, as it is concerned with user mobility and depends highly on the user's context, it poses multiple challenges to address. Second, we study how to determine a target application for a given query for unified mobile search where we also studied modeling contextual information available on mobile devices. Third, we address the conversational search task where we define an offline evaluation protocol for IR and build a dataset for researching the problem of asking clarifying questions for conversational search. Moreover, we proposed a neural approach that can retrieve and select clarifying questions to narrow down a user's search.

1.2 Thesis Outline

This thesis is organized into three parts and nine chapters. But first, **Chapter 2** reviews the related work on areas relevant to venue suggestion, mobile search, and conversational search, the three areas targeted by the thesis.

Part 1 focuses on venue suggestion where **Chapter 3** describes our work on studying various mobile relevance criteria in the context of the Text REtrieval Conference - Contextual Suggestion track (TREC-CS). The task was to retrieve and rank venues relevant to users' interest and context, given by a history of preferences in a mobile environment. Participants were provided with the same dataset built by NIST¹, and the result of their models was independently evaluated by TREC assessors. In TREC-CS 2015, we explored the linear combination of source, opinion, and category relevance criteria and were ranked as the best performing group. In 2016, we followed a more sophisticated approach and modeled complex contextual information. Our approach was again ranked as the best performing approach in TREC-CS 2016. Later, we expanded those works and studied various aspects of relevance criteria and their combination where we collected different datasets: (i) a large dataset of venue profiles on Foursquare²; (ii) a smaller dataset of venue profiles on Yelp³; (iii) a crowdsourced dataset of contextual features relevant to a trip; and (iv) a crowdsourced dataset of contextual labels for given features. We explore several similarity scores as part of a content-based recommendation model and the impact of several modalities of information on their performance. Furthermore, we move one step further and model user information need using a collaborative approach. To this end, in **Chapter 4**, we propose a two-step collaborative model that utilizes users' implicit feedback (i.e., check-in data) as well as contextual information. From the lessons we learned while experimenting content-based and collaborative relevance criteria estimation and combination, we found that the best approach would be to combine both strategies. Therefore, in **Chapter 5**, we propose a hybrid recommendation model outperforming all existing content-based and collaborative state-of-the-art models.

Although studying venue recommendation on mobile devices helped us acquire invaluable understanding and insight into how users' access information on their mobile devices, it missed an important element of mobile IR: search queries. **Part 2** elaborates on the work we have done on a unified mobile search. **Chapter 6** describes how we collected a dataset of mobile search queries that

¹the USA National Institute of Science and Technology

²<https://foursquare.com>

³<https://yelp.com>

were submitted to various applications. This study gave us interesting intuitions on how users seek information using various applications and helped us design a system for unified mobile search where we tackled, as the first step, the task of target apps selection. This task is concerned with returning a ranked list of applications for a given search query such that the applications that are ranked higher are more likely to address the user’s information need. **Chapter 7** presents our follow-up work where we collected a more realistic dataset of cross-app queries and captured mobile sensor data from users, where we modeled contextual information as part of a neural target apps selection model. Here we focused on the user’s interaction with the application as context and studied how the applications that a user interacts with in the past 24 hours help a system determine context more accurately.

Recently, there has been an increasing interest in conversational search systems, both on voice-only devices and smartphones. This motivated us to study conversational search on mobile devices in **Part 3**. To this aim, we studied the problem of asking clarifying questions for conversational search in **Chapter 8**. Asking questions is a crucial element of conversational systems mainly for two reasons: (i) systems are usually limited in the number of results they can present to users and (ii) conversation is the easiest means of interacting with users. Therefore, in cases where the user’s initial query is faceted, ambiguous, or incomplete, and the confidence of the system is low, the user’s intent can be clarified by asking relevant questions. We collected a large dataset of clarifying questions and their corresponding answers for various search scenarios. This helped us propose an offline evaluation framework and explore neural and non-neural question retrieval and selection models. The results showed that asking even only one clarifying question can lead to a retrieval improvement of over 170%.

Finally, **Chapter 9** concludes the thesis and describes various possible future directions that stem from this thesis.

1.3 Main Contributions

Here we describe the main contributions of each part of the thesis:

1. Venue Suggestion (Part 1):

- We introduce a set of relevance scores for measuring the similarity between a user’s history and a location considering location’s content and reviews.

- We present a probabilistic generative approach to finding the mapping between location taste keywords and user tags, thus modeling the personalized opinion of users about venues more accurately.
- We introduce a novel dataset for predicting contextually appropriate locations and show how to predict the contextually appropriate locations given the user's current context and evaluate its effectiveness on recommendation.
- We propose a general time-sensitive regularizer, taking into account the variance of users activities and venues popularity over time.
- We propose a novel two-phase CR-based POI recommendation algorithm incorporating users implicit check-in feedback with a focus on the top of the list.
- We introduce a novel CR framework with the focus on the top of the recommendation list while incorporating the cross-venue similarities into the model.
- We propose a simple yet effective hybrid recommendation system.

2. Unified Mobile Search (Part 2):

- We design and conduct two crowdsourcing tasks for collecting cross-app search queries for real-life search tasks.
- We present the first study of user behavior while searching with different apps as well as their search queries. In particular, we study the attributes of the search queries that are submitted to different apps and user behavior in terms of the apps they chose to complete a search task.
- We propose two neural models for target apps selection.
- We design and conduct an in situ mobile search study for collecting thousands of real-life cross-app queries.
- We present the first in situ analysis of cross-app queries and users' behavior as they search with different apps. More specifically, we study different attributes of cross-app mobile queries concerning their target apps, sessions, and contexts.
- We propose a context-aware neural model for target apps selection.
- We evaluate the performance of state-of-the-art retrieval models for this task and compare them against our proposed model.

3. Conversational Search (Part 3):

- We formulate the task of selecting and asking clarifying questions in open-domain information-seeking conversational systems.
- We propose an offline evaluation framework based on faceted and ambiguous queries and collect a novel dataset, building on top of the TREC Web Track 2009-2012 collections.
- We conduct oracle experiments and analyze the behavior of the model under various conditions.
- We propose a retrieval framework, consisting of three main components as follows: (i) question retrieval; (ii) question selection; and (iii) document retrieval.

1.4 Resources Created and Released

Throughout my Ph.D., we have collected and released multiple resources with the aim of foster research in relevant areas. Here we provide a brief overview of the resources we have created. We first describe the data collections, followed by the tools that we have developed.

- Data Collections:
 - **TREC-CS 2015-16 Venue Information:** This dataset contains the profiles of the venues on two main LBSNs. The venues are those that are present in the TREC-CS 2015 and 2016 datasets. The dataset contains over 300K venue profiles on Foursquare and 20K of venue profiles on Yelp. It contains rich meta and textual information about venues such as venue address, ratings, and reviews (see Section 3.5).
 - **Venue Appropriateness Feature and Labels:** We built this dataset via crowdsourcing. The aim is to assess the appropriateness of a venue for a given contextual description. We collected this dataset based on general contextual descriptors as well as venue categories. This makes the dataset general enough to be used on venue suggestion datasets (see Section 3.5).
 - **UniMobile:** We designed two crowdsourcing tasks to collect cross-app search queries paired with their relevant apps. The first task provided us with over 200 real-life mobile search tasks targeting various

modalities and topics. The second task collected thousands of query-app pairs for every mobile search tasks. We have released both search task definitions and query-app pairs (see Section 6.2).

- **ISTAS:** This is the first in situ collection of cross-app mobile search queries. We have collected thousands of query-app pairs via self-reporting of nearly 300 users over three months. We have also collected multiple sensor readings and app usage statistics (see Section 7.2).
 - **Qulac:** To enable offline evaluation of asking clarifying questions we have collected over 10K questions and answers via crowdsourcing. We collected the data in four steps involving both crowdsourcing and expert annotation. Qulac enables evaluation of models for asking clarifying questions (see Section 8.3).
- Tools:
 - **uSearch:** We developed a bespoke Android app, called uSearch, with which we collected the ISTAS data collection. uSearch is open source and can be used to collect various types of data that involve user behavior and search on mobile devices. It is designed to perform self-report data collection or user study. However, the architecture of uSearch enables the easy and quick extension for other tasks (see Section 7.2.1).
 - **Omicron:** We developed Omicron based on the infrastructure that we developed for uSearch. Omicron is designed for performing task-based user studies for mobile search. Also, we improved the design of its user interface as well as its efficiency (see [20]).

1.5 Publication Overview

The material of this thesis was published in conferences and journals as listed below:

- Chapter 3 is based on:
 - [11] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. User model enrichment for venue recommendation. In *Proceedings of the Asia Information Retrieval Societies Conference (AIRS)*, pages 212–223. Springer, 2016.

-
- [16] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. Personalized keyword boosting for venue suggestion based on multiple lbsns. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 291–303. Springer, 2017.
 - [4] Mohammad Aliannejadi and Fabio Crestani. Venue appropriateness prediction for personalized context-aware venue suggestion. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1177–1180. ACM, 2017.
 - [15] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. A cross-platform collection for contextual suggestion. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1269–1272. ACM, 2017.
 - [7] Mohammad Aliannejadi and Fabio Crestani. Personalized context-aware point of interest recommendation. *ACM Trans. Inf. Syst.*, 36(4): 45:1–45:28, 2018.
 - Chapter 4 is based on:
 - [22] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. A joint two-phase time-sensitive regularized collaborative ranking model for point of interest recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2019 (in press).
 - Chapter 5 is based on:
 - [17] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. A collaborative ranking model with multiple location-based similarities for venue suggestion. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 19–26. ACM, 2018.
 - Chapter 6 is based on:
 - [19] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Target apps selection: Towards a unified search framework for mobile devices. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 215–224. ACM, 2018.
 - Chapter 7 is based on:

- [18] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. In situ and context-aware target apps selection for unified mobile search. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1383–1392. ACM, 2018.
- Chapter 8 is based on:
 - [21] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 475–484. ACM, 2019.

1.6 Additional Publications

These additional papers were published in conferences, workshops, and evaluation forums during this thesis, but were not included in it to maintain the coherency of the thesis.

- [10] Mohammad Aliannejadi, Seyed Ali Bahrainian, Anastasia Giachanou, and Fabio Crestani. University of Lugano at TREC 2015: Contextual suggestion and temporal summarization tracks. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2015.
- [12] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. Venue appropriateness prediction for contextual suggestion. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2016.
- [14] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. Personalized ranking for context-aware venue suggestion. In *Proceedings of the Symposium on Applied Computing (SAC)*, pages 960–962. ACM, 2017.
- [13] Mohammad Aliannejadi, Maram Hasanain, Jiaxin Mao, Jaspreet Singh, Johanne R. Trippas, Hamed Zamani, and Laura Dietz. ACM SIGIR student liaison program. *SIGIR Forum*, 51(3):42–45, 2017.
- [5] Mohammad Aliannejadi and Fabio Crestani. A collaborative ranking model with contextual similarities for venue suggestion. In *Proceedings of the Italian Information Retrieval Workshop (IIR)*, 2018.

-
- [6] Mohammad Aliannejadi and Fabio Crestani. Venue suggestion using social-centric scores. *CoRR*, abs/1803.08354, 2018.
 - [20] Mohammad Aliannejadi, Morgan Harvey, Luca Costa, Matthew Pointon, and Fabio Crestani. Understanding mobile search task relevance and user behaviour in context. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 143–151. ACM, 2019.
 - [95] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. ANTIQUE: A non-factoid question answering benchmark. *CoRR*, abs/1905.08957, 2019.
 - [159] Hossein A. Rahmani, Mohammad Aliannejadi, Rasoul Mirzaei Zadeh, Mitra Baratchi, Mohsen Afsharchi, and Fabio Crestani. Category-aware location embedding for point-of-interest recommendation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 173–176. ACM, 2019.
 - [42] Hamed Bonab, Mohammad Aliannejadi, John Foley, and James Allan. Incorporating hierarchical domain information to disambiguate very short queries. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 51–54. ACM, 2019.
 - [158] Hossein A. Rahmani, Mohammad Aliannejadi, Sajad Ahmadian, Mitra Baratchi, Mohsen Afsharchi, and Fabio Crestani. LGLMF: local geographical based logistic matrix factorization model for POI recommendation. In *Proceedings of the Asia Information Retrieval Societies Conference (AIRS)*, 2019.

Chapter 2

Literature Review

In this chapter, we review the literature on the topics that we have covered in this thesis. In particular, the chapter consists of three sections, reviewing works related to venue suggestion, mobile IR, and conversational search, respectively.

2.1 Venue Suggestion

Recommender systems play an important role in satisfying users' expectations for many online services such as e-commerce, LBSN and social network websites. Venue Suggestion can be divided into three categories of approaches, namely, content-based, collaborative, and hybrid models.

Content-based approaches generally rely on modeling users behavior in the form of user profiles. These models also generate POI profiles to characterize venues' features and produce the recommendation by computing the similarity between users and venues profiles [3, 199]. Content-based filtering has several advantages such as being user independent and transparent, as well as being able to recommend items that have not yet been rated by any users [132]. The dependency on user and item profiles, however, introduces a number of drawbacks, namely, limited content analysis, over-specialization, and its inability to recommend items to a new user [132]. The need for domain knowledge and feature generation leads to limited content analysis. Moreover, such models highly depend on the generated profiles and users past behavior. Therefore, what a system recommends to a user is very similar to what they have visited in the past, limiting the serendipity of recommendation. Finally, the system is able to generate a user profile only after it receives sufficient ratings, limiting the ability of such approaches to handle the cold-start problem.

Collaborative filtering (CF) approaches are based on the core idea that users

with similar behavioral history tend to act similarly in the future [32, 88]. A large body of research has been done following this idea [84, 90, 203, 217]. CF can be divided into two categories: memory-based and model-based. Memory-based approaches consider user rating as a similarity measure between users or items [169]. Model-based approaches, on the other hand, employ techniques like matrix factorization [116]. CF approaches often suffer from data sparsity since there are a lot of available locations, and a single user can visit only a few of them. As a consequence, user-item matrix of CF becomes very sparse, leading to poor performance of recommender systems in cases that there is no significant association between users and items.

Many studies have tried to address the data sparsity problem of CF by incorporating additional information into the model [201, 208]. More specifically, Ye et al. [201] argued that users' check-in behavior is affected by the spatial influence of locations and proposed a unified location recommender system incorporating spatial and social influence to address the data sparsity problem. Yuan et al. [208], on the other hand, proposed a time-aware collaborative filtering approach. More specifically, they recommended locations to users at a particular time of the day by mining historical check-ins of users in LBSNs. Yin et al. [203] proposed a model which captures user interests as well as local preferences to recommend locations or events when users are visiting a new city. Ference et al. [84] took into consideration user preference, geographical proximity, and social influences for POI recommendation. Griesner et al. [90] also proposed an approach integrating temporal and geographic influences into matrix factorization. Zhang and Chow [216] aggregated ratings of users' friends as well as the bias of users on venue categories as power-law distributions. Reviews reveal the underlying reasons of users' ratings related to a particular location and that is why many researchers have tried to incorporate the review text into the recommendation algorithms. In fact, as argued by Chen et al. [54], online reviews significantly help a system to deal with the data sparsity problem. For instance, as showed by Zhang et al. [217], fusing virtual ratings derived from online reviews into CF improves the recommendation effectiveness. Hariri et al. [92] tried to predict user's context from their reviews about venues by learning a Labeled Latent Dirichlet Allocation (LDA) [41] model on a dataset from TripAdvisor and using the predicted contextual information to measure the relevance of a venue to a user. In fact, the effectiveness of topic models has been extensively experimented in other domains [34, 35, 136].

Another line of research lies in combining content-based and collaborative approaches, aiming to use the advantages of both models in various domains [103, 137, 168, 188] for improved recommendation. More specifically, [25] studied a

number of content-based, collaborative, and hybrid recommendation algorithms for contextual suggestion and concluded that hybrid methods perform best for this task, followed by content-based and collaborative approaches. As mentioned earlier, content-based modeling can effectively recommend POIs that have not been visited in the past by other users. This is specifically crucial for cases where the data is very sparse. The TREC Contextual Suggestion (TREC-CS) track is an example of such scenario in which a user is visiting a new town with no prior check-in data available in the dataset. That is why content-based approaches generally outperform collaborative approaches on this dataset. Moreover, since hybrid approaches are able to fuse information from both models and leverage each model's advantages, they perform best for various recommendation tasks such as TREC-CS.

In the remaining of this section, we review the literature on the topics relevant to the thesis. We start by reviewing research on contextual suggestion, followed by studies on context-aware recommendation. Finally, we focus on the topic of collaborative ranking.

2.1.1 TREC Contextual Suggestion

The TREC-CS track [96] aimed to encourage research on context-aware POI recommendation. In fact, the task was to produce a ranked list of locations for each user in a new city, given the user's context and history of preferences in 1-2 other cities. The contextual dimensions were the trip duration, the season, the trip type, and the type of group with whom the user was traveling. These contextual dimensions were introduced in TREC-CS 2015. Since then, among the top runs, few approaches tried to leverage such information. Arampatzis and Kalamatianos [25] studied the performance of various content-based, collaborative, and hybrid fusion methods on TREC-CS and they found that content-based methods performed best among these methods. Yang and Fang [198] introduced some handcrafted rules for filtering locations based on their appropriateness to a user's current context. According to them, applying such filters degrades the performance of the system. Hence, one can conclude that contextual appropriateness is not a simple problem of applying some deterministic rules to filter locations. Yang et al. [199] created rich user profiles aggregating online reviews from other users and measured the similarity between a new location and a user profile.

Yuan et al. [209] proposed to consider both geographical and temporal influences while recommending POIs to the users via a geographical-temporal influences aware graph. They proposed to propagate these influences using a breadth-first strategy. Cheng et al. [57] proposed a multi-center Gaussian model

to capture users' movement pattern as they assumed users' movements involves several centers. In a more recent work, Zhang et al. [215] considered three travel-related constraints (i.e., uncertain traveling time, diversity of the venues, and venue availability) and use them to prune the search space. Griesner et al. [90] also proposed an approach integrating temporal and geographic influences into matrix factorization. Cui et al. [75] investigated how geotagged photos can be linked to venues to study users' tastes. Finally, Yuan et al. [207] addressed the data sparsity problem assuming that users tend to rank higher the POIs that are geographically closer to the one that they have already visited.

2.1.2 Context-Aware POI Recommendation

Another line of research tries to leverage available contextual information to enhance the performance of a recommender system. Context-aware recommendation has been categorized into three types [1]: (i) pre-filtering: data selection is done based on context; (ii) post-filtering: recommendation is done using a traditional approach and context is used to filter venues; (iii) contextual modeling: contextual information is incorporated into the model. Our work aims at modeling the contextual information by re-ranking the recommendations. Adomavicius et al. [2] proposed a multidimensional context pre-filtering model based on the online analytical processing for decision support. Park et al. [146] computed a weighted sum of the conditional probabilities of restaurants' attribute values. They automatically detected users' physical contexts such as the time of the day, the position, and the weather and used a Bayesian network for expressing their probabilistic influences. Levi et al. [122] developed a weighted context-aware recommendation algorithm to address the cold start problem for hotel recommendation. More specifically, they defined context groups based on hotel reviews and followed a user's preferences in trip intent and hotel aspects as well as the user's similarity with other users (e.g., nationality). Other works focused on time as context [78, 83, 86, 208]. Gao et al. [86] developed a time-aware recommendation model. Fang et al. [83] proposed a model which takes into account both spatial and temporal context to address the data sparsity problem. Deveaud et al. [78] modeled locations popularity in the immediate future utilizing time series. They leveraged the model to make time-aware POI recommendation. Braunhofer et al. [45] used various complex contextual factors such as budget, companion, and crowdedness to overcome the cold start problem. They developed an active learning strategy and a context-aware recommendation algorithm using an extended matrix factorization model.

2.1.3 Collaborative Ranking

Collaborative ranking is done by combining the ideas of CF and Learning to Rank (LTR). LTR methods have been proven to be effective in Information Retrieval (IR) [128]. LTR learns a ranking function which can predict a relevance score given a query and document. There are three categories of LTR, namely, point wise [71], pair wise [47], and list wise [51]. In short, point-wise approaches predict ranking scores for individual items. Pair-wise approaches, on the other hand, learn the order of the items, comparing the rank position of pairs of items. List-wise approaches consider an entire ranked list of items as individual training example. CR takes the idea of predicting preference order of items from LTR and combines it with the idea of learning the loss function in a collaborative way [36]. Weimer et al. [189] used a surrogate convex upper bound of Normalized Discounted Cumulative Gain (nDCG) error together with matrix factorization as the basic rating predictor. Shi et al. [172] explored optimizing a surrogate lower bound for Expected Reciprocal Rank (ERR) for data with multiple levels of relevance.

Christakopoulou and Banerjee [60] followed the idea of pair-wise LTR approaches. In particular, the authors base their work on LTR methods with an emphasis on the top of the recommendation list. This approach, however, is limited to explicit user feedback such as user ratings for movies. Rafailidis and Crestani [156] presented an LTR model, taking into account the relevant items of users and their friends, pushing these items at the top of the list. Rendle et al. [165] presented a generic optimization criterion as well as a learning algorithm for incorporating implicit feedback while learning personal ranking for users, demonstrating its effectiveness on approaches such as matrix factorization. In Rafailidis and Crestani [154], authors combined various LTR methods into a joint model aiming to enhance the recommendation accuracy with trust relationships. In a more recent work, Rafailidis and Crestani [157] proposed a model considering not only relevant items of the user and her trusted friends, but also the items of her distrusted foes. Lee et al. [121] assumed that the user-item matrix is low rank within certain neighborhoods of the metric space and minimized a pair-wise loss function. Hu and Li [100] proposed a point-wise CR approach considering user ratings as ordinal rather than viewing them as real values or categorical labels. Also, they emphasized more on positively rated items to improve the performance at the top of recommended list.

2.1.4 Time-Aware Recommendation

Many researchers have studied temporal influence on users' preferences. A group of studies conducts time-aware recommendation learning of users' temporal preference for specific time slots and for recommending POIs for a given time slot, like the hour of a day [81]. Yuan et al. [208] computed the similarity between users by finding the same POIs at the same time slots in their check-in history to train a user-based CF model. Yao et al. [200] matched the temporal regularity of users with the popularity of POIs to improve a factorization-based algorithm. Li et al. [124] proposed a time-aware personalized model adopting a fourth-order tensor factorization-based ranking which enables the model to capture short-term and long-term preferences. Yin et al. [205] proposed a topic-region model that discovers the semantic, temporal, and spatial patterns of users' check-ins and uses the additional information to address the data-sparsity problem. Yin et al. [202] defined the temporal context as the public's attention at a certain time and proposed a temporal context-aware mixture model, modeling the topics related to users' interests and temporal context in a unified way. This work was later extended in [204] to a dynamic temporal context-aware mixture model, capturing users' evolving interests. Gao et al. [86] preserved the similarity of personal preference in consecutive time slots by considering different latent variables at each time slot for each user.

There also exists another category of approaches which tries to recommend the next POI to visit, known as successive POI recommendation. For example, Cheng et al. [56] captured sequential check-in behavior of users by training personalized Markov chains. Liu et al. [129] combined the ideas of both categories by recommending POIs for a particular time, exploiting sequential patterns of users.

2.2 Mobile Search

In this section, we review the work related to the second part of the thesis. While the study of unified mobile search is a new research area, it has roots in previous research related to mobile IR, human interaction with mobile devices (mobile HCI), federated, and aggregated search. Moreover, relevant research has been done in the area of proactive IR where a system aims to provide personalized information to users based on their context. Other relevant works can be found in the areas of query classification and neural networks. In the following, we summarize the related research in each of these areas.

2.2.1 Mobile IR

One of the main goals of mobile IR is to enable users to carry out all the classical IR operations using a mobile device [73]. One of the earliest studies on mobile IR was done by Kamvar and Baluja [105] where they did a large-scale mobile search query analysis. They also found that mobile queries were less diverse than desktop queries at the early days of mobile search. Crestani and Du [72] compared spoken and written queries in a lab study with 12 users. They found that spoken queries are longer and more similar to natural language queries. In another study, Church et al. [65] analysed six million search queries in a period of one week. They also analysed the click-thru rate and found that users focused on the first few search results. In fact, Song et al. [177] studied a commercial search log and found significant difference in search patterns done using iPhone, iPad, and desktop. For instance, they found that query length on mobile devices were longer. However, they suggested that the query length continued to change and this could be a sign of evolving mobile usage patterns. Also, query categories, usage time, and location of usage were different among different devices. In a similar study, Montanez et al. [139] studied search across multiple devices including smartphones. In a more recent study, Guy [91] analysed 500,000 spoken queries from a commercial mobile search app, submitted via a voice interface. The analysis confirmed that voice queries are longer on average and are closer to natural language. Moreover, they are more focused on multimedia content and require less interaction with the device's touchscreen.

More recently, research has been done on various topics in mobile IR such as app recommendation search [144, 145]. For instance, Shokouhi et al. [174] studied query reformulation patterns in mobile query logs and found that users do not tend to switch between voice and text while reformulating their queries. Park et al. [144] represented apps using online reviews for improved app search on the market. Williams et al. [192] leveraged mobile user gesture interactions, such as touch actions, to predict good search abandonment on mobile search. Park et al. [145] inferred users implicit intentions from social media for the task of app recommendation. Ong et al. [140] observed different user behaviors while doing mobile and desktop search as the amount of information scent was altered.

A few industrial systems exist aiming to provide users with unified mobile search. Apple Spotlight¹ is the most popular example of such systems that is available on iOS devices. Also, Sesame Shortcuts² is an Android app that creates easy-to-access shortcuts to the installed apps. The shortcuts are also accessible

¹[https://en.wikipedia.org/wiki/Spotlight_\(software\)](https://en.wikipedia.org/wiki/Spotlight_(software))

²<http://sesame.ninja/>

via keyword-based queries. Despite the existence of these systems, research on cross-app search has not yet been done.

2.2.2 Mobile HCI

Understanding human interaction while doing mobile search has become an area of interest since mobile devices are constantly evolving [177]. For this reason, many researchers have conducted user studies to understand various aspects of user behaviour and interaction in relation to mobile search. Sohn et al. [176] conducted a two-week diary study from 20 participants in which they found that contextual features such as activity and time influence 72% of mobile information needs. Kaikkonen [104] asked 390 mobile Internet users to fill an online survey, followed by a 23 face-to-face interviews. They analysed the impact of location and Web page design on mobile phone browsing behaviour. At the time of the study, they found that there were more female users from Asian countries and they preferred mobile tailored Web content over full Web content.

Church and Smyth [63] studied the intent behind mobile information needs of twenty users over four weeks via a diary study. They observed significant differences between mobile and desktop information needs. In particular, they found that users had many non-informational information needs, with geographical and personal information needs being popular. Later, Church and Oliver [62] carried out another diary and interview study to understand the shift of mobile information needs. The study was done over a four-week period with 18 active mobile users, discovering that the popularity of stationary mobile Web access was increasing. In another attempt to understand users' information needs, Church et al. [66] conducted a large-scale *snippet-based diary* [43] study with 100 participants throughout a three-month period. This technique allowed users to capture moments in situ and send them via SMS or MMS. Later, they could access a Web site in which they would review the messages and provide more details about their context. They found significant differences in terms of information needs and how they were addressed depending on users' gender, device, and location.

Also, the studies aimed to analyse touch-screen gestures [192], effect of searching on-the-go [93], effect of result snippets [112] as well as users' perception of result usefulness [135]. In particular, Williams et al. [192] conducted a lab study with 60 participants, focusing on the analysis of user gesture interactions, such as touch actions, and their relation with good search abandonment on mobile search. They showed that the time spent interacting with answers on a SERP is positively correlated with good abandonment and satisfaction. Through another lab study consisting of 72 participants, Ong et al. [140] observed different pat-

terns in user behaviour while doing mobile and desktop search as the amount of information scent was altered. They found that users' behaviour differ significantly in a mobile environment. For instance, desktop users preferred SERPs with a higher number of relevant search results; whereas this preference was not observed on the mobile environment. Moreover, Harvey and Pointon [93] recruited 24 participants and did a lab study where they found that fragmented attention of users while searching on-the-go, affects their search objective and performance perception. Kim et al. [112] conducted a lab study of 24 participants and analysed the effect of result snippet size on mobile search time and accuracy. They found that, for informational tasks on mobile devices, longer snippets lead to longer search times with no better search accuracy. In fact, they conclude that the optimum mobile snippet length is two to three lines. Later, Mao et al. [135] investigated result usefulness in mobile search via a lab study involving 43 participants and confirmed that usefulness feedback can better reflect user satisfaction than relevance annotations.

Moreover, as mobile devices evolved to become our main means of accessing data, researchers tried to understand the impact of mobile applications (apps) on Web search [53] as well as search within apps [18, 19]. In this context, Carrascal and Church [53] ran an in situ study with 18 users, analysing how users interacted with apps while they were doing mobile Web search. They showed that users' interactions with apps have an impact on search. For example, they found significant differences in the categories of apps used within search sessions as opposed to non search sessions.

2.2.3 Context-Aware Search

Most of the previous work in context-aware search is based on the user's search history [171, 190, 194]. Shen et al. [171] presented context-sensitive language models based on users' short-term search history. White et al. [190] investigated ways to optimally combine the query and its context by learning a model that predicts the context weight for each query. Bennett et al. [39] estimated the location preference of a document and used it to improve Web search.

2.2.4 Proactive IR

The aim of proactive IR systems is to anticipate users' information needs and proactively present information cards to them. Shokouhi and Guo [173] analyzed user interactions with information cards and found that the usage patterns of the cards depend on time, location, and user's reactive search history. Benetka

et al. [38] showed that information needs vary across activities as well as during the course of an activity. They proposed a method to leverage users' check-in activity for recommending information cards. Sun et al. [180] proposed a collaborative nowcasting model, tackling the intent monitoring problem, utilizing the collaborative capabilities among users. This thesis focuses on the queries that users issue in different apps. Queries can express complex information needs that are impossible to infer from context.

2.2.5 Federated and Aggregated Search

A unified mobile search system distributes a search query to a limited number of apps that it finds more relevant to a search query. There is a considerable overlap between the target apps selection task and federated/aggregated search. In federated search, the query is distributed among uncooperative resources with homogeneous data; whereas in aggregated search, the content is blended from cooperative resources with heterogeneous data [26]. Given the uncooperative environment of most federated search systems, Callan and Connell [48] proposed a query-based sampling approach to *probe* various resource providers and modeled them based on the returned results. In most aggregated search systems, on the other hand, different resources are parts of a bigger search system and thus cooperative. Moreover, an aggregated search system can even access other metadata such as users' queries and current traffic [26]. Diaz [80] proposed modeling the query dynamics and collection to detect news queries for integrating the news *vertical* into the result page. This work was later extended by Arguello et al. [27] to include images, videos, and travel information. In this thesis, we assume an uncooperative environment because the contents of apps are not accessible to the unified search system. Moreover, given the existence of various content types in different apps, we assume documents to be heterogeneous.

2.2.6 Query Classification

Our work is also related to research in query classification where different strategies are used to assign a query to predefined categories. Kang and Kim [107] defined three types of queries arguing that search engines require different strategies to deal with queries belonging to each of the classes. Shen et al. [170] introduced an intermediate taxonomy to classify queries to specified target categories. Cao et al. [49] leveraged conditional random fields to incorporate users' neighboring queries in a session as context. More recently, Zamani and Croft [210]

studied word embedding vectors for the query classification task and proposed a formal model for query embedding estimation.

2.3 Conversational Search

Finally, we review the work related to the third part of the thesis, that is, conversational search. Our work is mainly related to the areas of conversational IR, clarifying questions, and conversational question answering. In the following, we briefly summarize the related research in each of these areas.

2.3.1 Conversational IR

While conversational search has roots in early IR research, the recent advances in automatic voice recognition and conversational agents have created increasing interest in this area.

One of the first works in conversational IR dates back to 1987 when Croft and Thompson [74] proposed I³R that acted as an expert intermediary system, communicating with the user in a search session. A few years later Belkin et al. [37] characterized information-seeking strategies for conversational IR, offering users choices in a search session based on case-based reasoning. Since then researchers in the fields of IR and NLP have studied various aspects of this problem. Early works focused on rule-based conversational systems [186, 191], while another line of research investigated spoken language understanding approaches [8, 9, 98, 149] for intelligent dialogue agents in the domain of flight [99] and train trip information [28]. The challenge was to understand the user's request and query a database of flight or train schedule information accordingly. The recent advances of conversational agents have attracted research in various aspects of conversational information access [19, 29, 30, 40, 181, 195]. One line of research analyzes data to understand how users interact with voice-only systems [178]. Radlinski and Craswell [153] proposed a theoretical framework for conversational search highlighting the need for multi-turn interactions with users for narrowing down their specific information needs. Also, Trippas et al. [183] studied conversations of real users to identify the commonly-used interactions and inform the design of a conversational search system. Moreover, research on query suggestion is relevant to our work if we consider suggesting queries as a means of clarifying users' intent in a traditional IR setting [153]. Result diversification and personalization is one of the key components for query suggestion [102], especially when applied to small-screen devices. In particular, Kato and Tanaka

[109] found that presenting results for one facet and suggesting queries for other facets is more effective on such devices.

2.3.2 Clarifying Questions

Research on clarifying questions has attracted considerable attention in the fields of NLP and IR. People have studied human-generated dialogues on question answering (QA) websites, analyzing the intent of each utterance [152] and, more specifically, clarifying questions [44]. Kiesel et al. [111] studied the impact of voice query clarification on user satisfaction and found that users like to be prompted for clarification. Much work has been done on interacting with users for recommendation. For instance, Christakopoulou et al. [61] designed a system that can interact with users to collect more detailed information about their preferences in venue recommendation. Also, Sun and Zhang [181] utilized a semi-structured user query with facet-value pairs to represent a conversation history and proposed a deep reinforcement learning framework to build a personalized conversational recommender system. Focusing on clarifying questions, Zhang et al. [218] automatically extracted facet-value pairs from product reviews and considered them as questions and answers. They proposed a multi-memory network to ask questions for improved e-commerce recommendation. Our work is distinguished from these studies by formulating the problem of asking clarifying questions in an open-domain information-seeking conversational setting where several challenges regarding extracting topic facets [114] are different from a recommendation setting.

In the field of NLP, researchers have worked on question ranking [161] and generation [162, 187] for conversation. These studies rely on large amount of data from industrial chatbots [151, 187], query logs [164], and QA websites [161, 162, 182]. For instance, Rao and Daumé [161] proposed a neural model for question selection on a simulated dataset of clarifying questions and answers extracted from QA websites such as StackOverflow. Later, they proposed an adversarial training for generating clarifying questions for a given product description on Amazon [162]. Also, Wang et al. [187] studied the task of question generation for an industrial chatbot.

2.3.3 Conversational Question Answering

Several NLP areas fall into this category of research such as question answering and language understanding. SQuAD [160] is a corpus on reading comprehension consisting of questions and answers about various topics. CoQA [163] ad-

dresses the challenge of conversational question answering. While Choi et al. [59] extended the idea of CoQA data collection with the difference that the person asking questions did not see the description of information and called it QuAC. Both CoQA and QuAC paired two crowdworkers to participate in a live conversation.

Part I

Venue Suggestion

Chapter 3

Content-based User Modeling for Venue Suggestion

3.1 Introduction

Despite the ever-growing number of users on the major Location-Based Social Networks (LBSNs) with long history of check-ins and interactions, these platforms are still facing many challenges that need to be addressed. As a consequence, a great deal of research is being carried out on improving user and POI profiles on LBSNs. As the time goes by and the history of users as well as POIs becomes richer, these platforms should be able to take advantage of the massive amount of information they are exposed to. This provides a unique opportunity to study how users' behavior is influenced by various factors that occur in a long span of time. Apart from personal preference and interest, the user behavior is influenced and, in many cases, constrained by local and contextual preferences [83]. For instance, a user may be a big fan of *nightlife spots*. However, when traveling with their family, they may prefer not to visit such locations. Hence, it is crucial to consider a user's context when recommending locations to them. It is also important to note that the user's context often introduces new constraints, not necessarily in tune with their opinion and interest. To this end, the main focus of the Text REtrieval Conference (TREC) Contextual Suggestion (TREC-CS) track¹ in 2015 [76] and 2016 [96] was to improve location recommendation with the aid of contextual information. However, not many successful participants took into account context in their proposed approaches. Thus, applying contextual constraints still remains a challenge for context-aware

¹<https://sites.google.com/site/trecontext/>

POI recommendation.

Given the easy access to the Internet and availability of mobile devices such as cell phones, smart watches and tablets, users tend to leave their check-in data more often. However, writing a long review on such devices is not as trivial as is using a desktop computer. As a consequence, the majority of users rate locations without writing a review. Reviews contain a wealth of information relevant to the user's opinion and view about a location; for example a user's opinion about a location's view or staff. In order to compensate for the absence of such information, an LBSN could assist a user with a few related predefined *tags*, from which the user can conveniently select those expressing their opinion. Predefined tags come very handy especially on smaller devices such as smart watches enabling users to express themselves with the aid of a couple of taps. Modeling users with such tags is very challenging since user tags are much more sparse compared to user ratings. Thus the traditional CF approach could not be applied for user tag modeling. Furthermore, in real-world POI recommendation scenarios for mobile devices, the top 10 locations are usually interesting to users [57], because of the screen size of a typical mobile device and limited effort a typical user spends to go through the recommendation list. Therefore, providing a personalized ranking to the user is crucial, making this task a *top k recommendation* task.

In this chapter, we present our initial attempt to model users' information need and context. As a first step, we followed a content-based approach for modeling users and POIs. To this aim, our contributions can be summarized as follows:

1. We introduce a set of relevance scores for measuring the similarity between a user's history and a location considering location's content and reviews.
2. We present a probabilistic generative approach to find the mapping between location taste keywords and user tags thus modeling the personalized opinion of users about venues more accurately.
3. We address the sparsity problem by performing personalized boosting of location keywords in a user's history.
4. We explore different machine learning models to predict user tags and evaluate their effectiveness in terms of both tag prediction and recommendation effectiveness.
5. We introduce a brand new dataset for predicting contextually appropriate locations and show how to do this given the user's current context and evaluate its effectiveness on recommendation.

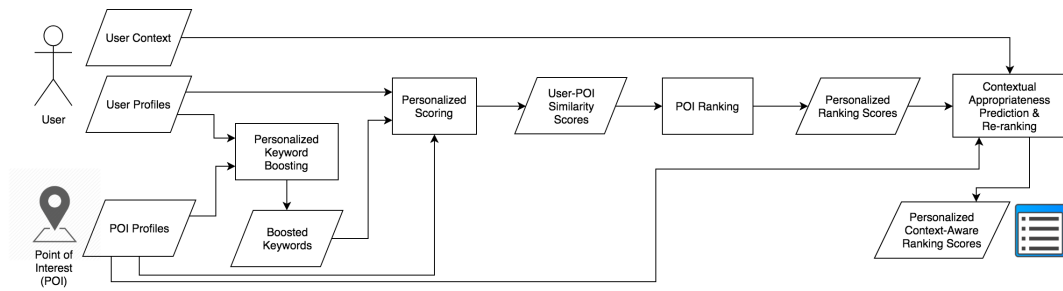


Figure 3.1. Overview of the proposed method.

6. We evaluate several learning to rank techniques to incorporate boosting and tag prediction into our POI recommendation model using information from multiple LBSNs.

Figure 3.1 shows an overview of our proposed method. In the first step, user and POI profiles are analyzed to perform personalized keyword boosting, resulting in a list of boosted keywords for each user (see Section 3.2). Then the list of boosted keywords together with user and POI profiles are fed to the personalized scoring component to calculate the similarity scores between a given user and a POI. The scores are then passed to the ranking model (i.e., learning to rank) to produce a personalized ranked list of POIs for each user (see Section 3.4). Finally, given the user’s current context, the level of contextual appropriateness of every venue is predicted (see Section 3.3) and used to re-rank the personalized ranking scores, resulting in a personalized context-aware ranked list of POIs.

In fact, experiments show that combining multimodal information from multiple LBSNs improves POI recommendation significantly. Moreover, we show that the proposed mapping of location keywords to user tags enables us to predict user tagging behavior effectively. We also show that predicting contextually appropriate locations and reranking suggestions according to their contextual appropriateness results in a more accurate top k venue recommendation.

The remainder of the chapter is organized as follows. We begin with describing our methodology for crawling data from major LBSNs followed by two crowdsourcing tasks that we designed to collect contextual features in Section 3.5. Then, before explaining our proposed recommendation approach, we describe two critical components of our model in Sections 3.2 and 3.3. We first describe how we model the statistical mapping between user tags and venue taste keywords in Section 3.2.1. Then, we explore two directions to use this information. First, we explain how we use the computed mapping to reduce the dimensionality of venue taste keywords in Section 3.2.3. Second, we describe how we use

the computed mapping as training data to learn the sequential tagging of user tags in Section 3.2.4. The trained tagging models are then used to predict user tags for unseen venues. Section 3.3 elaborates our proposed approach to predict contextual appropriateness of locations. After describing the two main components of our model, in Section 3.4 we describe how we integrate them with other similarity measures in order to recommend POIs to users. Section 3.6 describes the evaluation protocol and Section 3.7 presents our experimental evaluation. We summarize this chapter in Section 3.8, describing the main findings and the need to explore this problem further as a collaborative approach.

3.2 Personalized Keyword Boosting

In this section we propose a probabilistic approach by which we map location keywords to user tags. Furthermore we propose two possible approaches to utilizing such additional information in order to enhance location recommendation. First, we propose to use the mapping as additional information to reduce the dimensionality of location keyword space in order to address the data sparsity problem. Second, we use the mapping to train a sequence labeling model to predict user tags for a new location. We use the outcome of both approaches to estimate the similarity of a location to a user in Section 3.4.

3.2.1 Personalized Keyword-Tag Mapping

In this section we present a probabilistic approach to map location keywords to user tags. We aim to find a meaningful correlation between the location content (e.g., keywords) and user tags since users annotate locations with tags based on both their personal views and locations' characteristics. We assume that the key characteristics that trigger the user's mind to annotate a location with a specific tag are of those listed in the location keywords. For example, tagging a location as *healthy-food* is a result of the user's personal view reflected in the location's characteristics (e.g., keywords). Hence, a user who believes *vegan* foods are healthy may tag a *vegan* location as *healthy-food*, whereas another user with a different view may tag a *sushi* place as *healthy-food*. Therefore, user tags are dependent on both users views and locations' characteristics. That is why we need to find a meaningful mapping between user tags and location keywords to take into account locations' characteristics. The mapping needs to be personalized to model users' personal views. Figure 3.2 depicts a real example mapping with a set of two user tags and four location keywords. Our ultimate goal is to deter-

mine the most likely mapping of location keywords to user tags, personalized for each user.

For a given user u , let $\mathbf{f}^I = \langle f_1 \dots f_j \dots f_J \rangle$ be a sequence of location keywords. We aim to find the sequence of user tags $\mathbf{t}^I = \langle t_1 \dots t_i \dots t_I \rangle$. Note that \mathbf{t}^I refers to a sequence named \mathbf{t} with the length of I . Hence, \mathbf{t}^i denotes a set with the size of i ($\mathbf{t}^i = \langle t_1 \dots t_i \rangle$), whereas t_i refers to the i -th item of a given sequence. Our aim is to find a user tag sequence maximizing $Pr(\mathbf{t}^I | \mathbf{f}^I)$:

$$\hat{\mathbf{t}}^I = \underset{\mathbf{t}^I}{\operatorname{argmax}} \{Pr(\mathbf{t}^I | \mathbf{f}^I)\} = \underset{\mathbf{t}^I}{\operatorname{argmax}} \{Pr(\mathbf{f}^I | \mathbf{t}^I) Pr(\mathbf{t}^I)\}, \quad (3.1)$$

where $Pr(\mathbf{t}^I)$ models user tags. In fact, given \mathbf{t}^I , this function determines to what extent \mathbf{t}^I is likely to be generated by a specific user. It basically models the user's behavior of tag annotation regardless of location keywords. We fairly assume that users annotate locations with a specific tag independent of other tags. In other words, we assume zero-order dependence of user tags. Hence, we rewrite $Pr(\mathbf{t}^I)$ as follows:

$$Pr(\mathbf{t}^I) = p(I) \prod_{i=1}^I p(t_i | \mathbf{t}^{i-1}, I) = p(I) \prod_{i=1}^I p(t_i | I), \quad (3.2)$$

where $\mathbf{t}^{i-1} = \langle t_1 \dots t_{i-1} \rangle$.

$Pr(\mathbf{f}^I | \mathbf{t}^I)$ in (3.1) models location keywords given a sequence of user tags \mathbf{t}^I . We need to find the optimum mapping between location keywords and user tags to optimally model location keywords given \mathbf{t}^I . Therefore, we marginalize the probability $Pr(\mathbf{f}^I | \mathbf{t}^I)$ over \mathbf{m}^J . We introduce \mathbf{m}^J as the latent variable defining how location keywords are mapped to user tags: $\mathbf{m}^J = \langle m_1 \dots m_j \dots m_J \rangle$, with $m_j \in \{1, \dots, I\}$:

$$Pr(\mathbf{f}^I | \mathbf{t}^I) = \sum_{\mathbf{m}^J} Pr(\mathbf{f}^I, \mathbf{m}^J | \mathbf{t}^I), \quad (3.3)$$

where

$$\begin{aligned} Pr(\mathbf{f}^I, \mathbf{m}^J | \mathbf{t}^I) &= p(\mathbf{m}^J | \mathbf{t}^I, I, J) p(\mathbf{f}^I | \mathbf{m}^J, \mathbf{t}^I, I, J) \\ &= p(J | \mathbf{t}^I) \prod_{j=1}^J [p(m_j | \mathbf{m}^{j-1}, J, \mathbf{t}^I, I) p(f_j | \mathbf{f}^{j-1}, \mathbf{m}^J, J, \mathbf{t}^I, I)], \end{aligned} \quad (3.4)$$

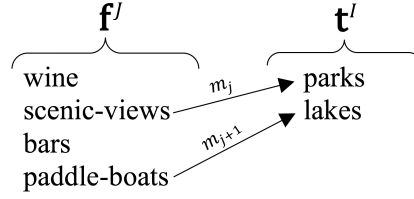


Figure 3.2. An example of mapping of $J = 4$ location keywords to $I = 2$ user tags.

where $\mathbf{m}^{i-1} = \langle m_1 \dots m_{i-1} \rangle$. We also assume a zero-order dependence for both m_j 's and f_j 's. Note that given the limited amount of data and its sparsity, we make some assumptions in order to reduce the number of parameters. Therefore, we consider $p(J|\mathbf{t}^I)$ only dependent on J and m_j is only dependent on the length of the user tag sequence I . We also assume that f_j depends only on t_{m_j} , i.e., the user tag associated to f_j according to the mapping. Notice that since \mathbf{f}^J denotes the sequence of location keywords of size J and \mathbf{t}^I denotes the sequence of user tags of size I , therefore $Pr(\mathbf{f}^J, \mathbf{m}^J | \mathbf{t}^I)$ also depends on the length of both sequences. Consequently, (3.4) is simplified as follows:

$$Pr(\mathbf{f}^J | \mathbf{t}^I) = p(J) \sum_{\mathbf{m}^J} \prod_{j=1}^J p(m_j | I) p(f_j | t_{m_j}). \quad (3.5)$$

3.2.2 Parameter Estimation Based on Expectation-Maximization

Assume that we have N pairs of training samples as in $S = \{(\mathbf{f}_{(1)}, \mathbf{t}_{(1)}), \dots, (\mathbf{f}_{(n)}, \mathbf{t}_{(n)}), \dots, (\mathbf{f}_{(N)}, \mathbf{t}_{(N)})\}$, the log-likelihood function for the training samples would be:

$$F(\vartheta) = \sum_{n=1}^N \sum_{j=1}^{J_n} \log \sum_{i=0}^{I_n} p(i | I_n) p(f_{jn} | t_{is}), \quad (3.6)$$

where $\vartheta := \{p(i|I), p(f|t)\}$ are the free parameters. To solve the parameter estimation problem of (3.6), we follow the Maximum Likelihood (ML) criterion subject to the constraint $\sum_f p(f|t) = 1$, for each user tag t . We use Lagrange multipliers to make the optimization problem unconstrained. However, since we introduced hidden variables into our model (Equation (3.3)), there is no closed-form solution to this optimization problem. Therefore, we follow the iterative process of the *Expectation-Maximization* (EM) algorithm.

To be able to follow the EM algorithm, we first define $Q(\vartheta, \hat{\vartheta})$ as:

$$\begin{aligned}
Q(\vartheta, \hat{\vartheta}) &= Q(\{p(i|I), p(f|t)\}; \{\hat{p}(i|I), \hat{p}(f|t)\}) \\
&= \sum_{n=1}^N \sum_{j=1}^{J_n} \sum_{i=0}^{I_n} \gamma_n(i|j, J_n, I_n) \log \{\hat{p}(i|I_n) \hat{p}(f_{j_n}|t_{i_n})\} \\
&= \sum_{n=1}^N \sum_{j=1}^{J_n} \sum_{i=0}^{I_n} \frac{p(i|I_n) p(f_{j_n}|t_{i_n})}{\sum_{i'=0}^{I_n} p(i'|I_n) p(f_{j_n}|t_{i'_n})} \log \{\hat{p}(i|I_n) \hat{p}(f_{j_n}|t_{i_n})\},
\end{aligned} \tag{3.7}$$

where $\gamma_n(i|j, J_n, I_n)$ is the posterior probability, defined as:

$$\gamma_n(i|j, J_n, I_n) = \frac{p(i|I_n) p(f_{j_n}|t_{i_n})}{\sum_{i'=0}^{I_n} p(i'|I_n) p(f_{j_n}|t_{i'_n})}.$$

According to the EM algorithm, we follow an iterative procedure for parameter estimation. After defining the relative objective function, $Q(\vartheta, \hat{\vartheta})$, we follow the usual steps of the algorithm:

1. E-step: calculate $Q(\vartheta, \hat{\vartheta})$ for all training samples in S with the previous estimate of ϑ .
2. M-step: optimize $Q(\vartheta, \hat{\vartheta})$ over $\hat{\vartheta}$.

We start the algorithm with uniform values for the parameters and follow the EM steps until convergence.

In the following we describe two possible directions to use the computed mapping.

3.2.3 Location Keywords Boosting

After finding the optimum mapping between the user tags and location keywords, we aim to use this additional knowledge in our system to address the sparsity problem and eventually enhance the recommendation performance. Take Figure 3.3 as a real example of such mapping from our dataset. As we can see, 19 taste keywords from one location are illustrated together with tags for the same location by 3 different users. Not surprisingly, the 3 sets have some tags in common such as “beer” and “cocktails.” However, each user has her own personal opinion and therefore her personal set of tags. The lines and the numbers in parentheses represent the result of our proposed mapping for these 3 users. Every mapped item is based on the user personal preference and behavior with respect to all locations in her history. As an example, we take one of the user tags

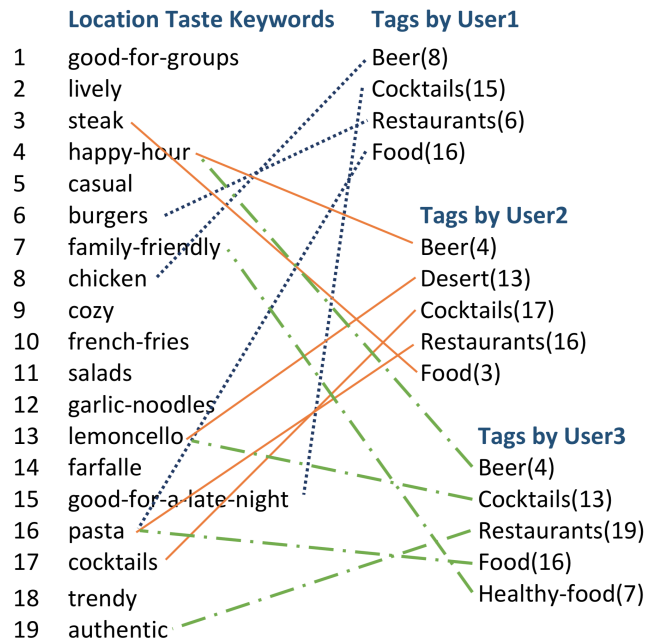


Figure 3.3. A sample of tags from three different users assigned to one single location and the calculated mapping. The lines connect each user tag to their mapped location keywords. Also, the index number of the mapped location keywords is written in parentheses for more convenient reading.

that is common between the three users: “cocktails.” What is interesting about this tag is that each user maps it to a different location keyword. For User1 “cocktails” is mapped to “good-for-a-late-night,” for User2 to “cocktails” and for User3 to “lemoncello.” All three location keywords are good candidates to be mapped to “cocktails” user tag, however, as we argued *each user has her own reasons to tag the same location with a different tag.*

After the observation of Figure 3.3, we assume that among the I location keywords, we can determine J keywords that are mapped to user tags and presumably are more interesting to the user. As in the example of Figure 3.3, the number of location keywords ($I = 19$) is much higher than user tags ($J_{user1} = 4$). Therefore, by boosting the mapped location keywords in our model we achieve two main goals: 1) reduce the location keyword space dimensions drastically (e.g., $19 \rightarrow 4$) and; 2) use the valuable information given by the users to detect those location keywords that are more interesting to each users.

Formally, let $\mathbf{f}^l = \langle f_1 \dots f_I \rangle$ be the set of keywords of a location and $\hat{\mathbf{f}} \in \mathbf{f}^l$ be the set of location keywords which are mapped to user tags. According to the result of our probabilistic mapping, we assume that there is a strong correlation

between $\hat{\mathbf{f}}$ and the user's interest. In other words, the keywords in $\hat{\mathbf{f}}$ correlate more to the user's interest as opposed to the other ones in \mathbf{f}^l . Hence, we boost $\hat{\mathbf{f}}$ to model the user's interests, reducing the data dimensionality from I to $|\hat{\mathbf{f}}|$. This helps us to address the data sparsity problem. The personalized boosted location keywords are used for POI recommendation (see Section 3.4).

3.2.4 User Tag Prediction

As an alternative approach, we explore three models to predict user tags. We utilize the result of our mapping model (\mathbf{m}) between location keywords and user tags to train a model able to predict user tags for a new location. We predict user tags for an unseen location as an alternative to keyword boosting. We explore this direction for two reasons: 1) to see how we can predict a user behavior in terms of tag annotation and; 2) to compare the effect of user tag prediction against location keyword boosting to see which strategy is able to enhance the recommendation more effectively. We follow two approaches to predict user tags: 1) we use the maximum likelihood criterion with our estimated parameters to generate the most likely set of user tags given a set of location keywords and; 2) we model the user tag prediction as a sequence labeling problem enabling us to apply different sequence labeling models.

Maximum likelihood. Here we describe how we follow Maximum Likelihood to leverage the learned mapping parameters in order to predict user tags for a new POI. As we mentioned in relation to (3.1) in Section 3.2.1, given a set of location keywords from Foursquare, \mathbf{f}^l , we aim to compute the most probable set of user tags, $\hat{\mathbf{t}}^l$. Once the model parameters are estimated using EM (see Section 3.2.2), one approach to predict user tags given a set of location keywords is to follow the maximum likelihood criterion (see (3.1)) to generate the most likely set of user tags. We search the space of user tag probabilities to find the optimum sequence of user tags following a Viterbi-like algorithm.

Sequence labeling. In the following, we begin with explaining how we model user tag prediction as a sequence labeling problem. Furthermore, we introduce the set of features we choose to train the tagging models as well as the tagging models we adopt. Assuming we have N sample mapped pairs of user tags and location keywords for each user: $S = \{(\mathbf{f}_{(1)}, \mathbf{t}_{(1)}), \dots, (\mathbf{f}_{(n)}, \mathbf{t}_{(n)}), \dots, (\mathbf{f}_{(N)}, \mathbf{t}_{(N)})\}$ with N corresponding mappings. That is, $M = \{\mathbf{m}_{(1)}, \dots, \mathbf{m}_{(n)}, \dots, \mathbf{m}_{(N)}\}$. We should model the tag prediction problem as a sequence labeling problem: given a sequence of location keywords we aim to predict the most probable sequence of user tags. In order to do this, we need to adapt the form of the training data.

As in a general sequence labeling problem, we need to assign a label from the target space to each item in the source space. Therefore, we should assign a label to all location keywords even if they are not mapped to any user tag. To this end, we automatically annotate location keywords following these steps:

1. For each $f_i \in \mathbf{f}_{(n)}$ mapped to a user tag with $\mathbf{m}_{(n)}$, we annotate f_i with its corresponding user tag m_j .
2. For each $f_i \in \mathbf{f}_{(n)}$ not mapped to a user tag with $\mathbf{m}_{(n)}$, we annotate f_i with “null.”

As for the example of Figure 3.3, a sample training sequence would be as follows:

$\underbrace{\text{burgers}}_{\text{restaurants}}, \underbrace{\text{chicken}}_{\text{beer}}, \underbrace{\text{good-for-a-late-night}}_{\text{cocktails}}, \underbrace{\text{pasta}}_{\text{food}}, \underbrace{\text{good-for-groups}}_{\text{null}}, \underbrace{\text{lively}}_{\text{null}}, \underbrace{\text{steak}}_{\text{null}}, \dots$

As we can see, each of the location keywords is annotated with a tag. Therefore, it is straightforward to use this data as training samples for a sequence tagger. We adopt two models as taggers: *Conditional Random Fields* (CRF) [119] and another tagger based on *Support Vector Machines* (SVM) [117]. The main advantage of these models is that they are discriminative [110]. Discriminative tagging models have proven to be more effective for sequence labeling mainly because they normalize the model over the whole training set, resulting in a more generalized model. Furthermore, discriminative tagging models accept a wider range of features, something that is of great importance to some applications.

As features of the sequence labeling models, for a user tag at position j , t_j , we only consider the location keyword at the same position, f_j . That is to follow our assumption that user tags only depend on one location keyword and since we assume that user tags are independent, we use a zero-order tagger model.

In this section, we first introduced a probabilistic framework for finding the mapping between location content (i.e., keywords) and user-annotated tags. Then, we described how this information can be leveraged to reduce the dimensionality of location keywords and hence to address the data sparsity problem. We also explored modeling this problem as a sequence labeling problem and used some state-of-the-art techniques to predict user tags given a new POI’s keywords. In Section 3.4, we show how we use the two mentioned directions to enhance POI recommendation.

Table 3.1. Description of different contextual information dimensions.

Context	Value	Short Reference	Description
Duration	Day Trip	day-trip	The duration of the trip is one day.
	Night Out	night-trip	The duration of the trip is a night out.
	Weekend Trip	weekend-trip	The trip lasts for a weekend.
	Longer	longer-trip	The trip lasts longer than a weekend.
Group	Alone	alone	The person travels alone.
	Friends	with-friends	The person travels with her friends.
	Family	with-family	The person travels with her family.
	Other	with-others	The person travels with a group. other than family and friends.
Type	Business	business-trip	The type of the trip is business.
	Holiday	holiday-trip	The type of the trip is holiday.
	Other	other-trip	The type of the trip is other than holiday and business, e.g., medical.

Table 3.2. Examples of contextual features generated using crowdsourcing.

Category	Context	$F_{\text{app}}(\text{cat}, \text{cxt})$
Beach	Trip type: Holiday	1.0
College & University	Trip duration: Weekend	-1.0
Shop & Service	Trip type: Holiday	0.71
Museum	Trip type: Business	-0.66
Pet Store	Trip duration: Weekend	-0.18
Medical Center	Trip type: Other	0.0

3.3 Contextual Appropriateness Prediction

In this section, we first define the problem of predicting the contextual relevance of locations. Then we present the set of features that we use to train the appropriateness classifier and introduce the dataset that we collected to train the classifier. The computed contextual relevance scores are then used to re-rank a ranked list of POIs in Section 3.4.

Let $V = \{v_1, \dots, v_n\}$ be a set of locations and $C_x = \{cx_1, \dots, cx_m\}$ a set of contextual descriptors. Our aim is to predict whether it is appropriate for a user to visit a location $v_i \in V$ under a given context C_x . Different contextual dimensions define user’s preferences, constraints, or requirements and are listed as follows:

Trip type (holiday, business, other), **Trip duration** (day trip, night out, weekend trip, longer) and **Group type** (alone, family, friends, other). More information could be found in Table 3.1. We model the problem as binary classification considering location categories and contextual descriptors as classification features.

3.3.1 Contextual Features

In this section, we describe the features that we used to train the appropriateness classifier. The degrees of appropriateness between location categories and contextual descriptors constitute our features.

We define a contextual feature function as follows:

Definition 3.3.1. A **contextual feature**, $F_{\text{app}}(cat, cxt)$, is a function determining the relevance of a POI category, cat , to a contextual dimension, cxt . $F_{\text{app}}(cat, cxt)$ ranges between -1 and $+1$ with -1 representing absolute inappropriateness and $+1$ absolute appropriateness.

For instance, assume a user wants to visit a location with category *nightlife-spot*, and her context is described as follows: holiday-trip, with-family, weekend-trip. The three features of this example are the appropriateness value between the location category and each of the contextual dimensions. Therefore, the features are $F_{\text{app}}(\text{nightlife-spot}, \text{holiday-trip})$, $F_{\text{app}}(\text{nightlife-spot}, \text{with-family})$, and $F_{\text{app}}(\text{nightlife-spot}, \text{weekend-trip})$.

In many cases, determining such contextual features is intuitive and can be done by one human annotator. However, there are several features that human annotators cannot agree on, like for example $F_{\text{app}}(\text{office}, \text{with-friends})$, $F_{\text{app}}(\text{food-and-drink-shop}, \text{business-trip})$, or $F_{\text{app}}(\text{stadium}, \text{night-out-trip})$. Hence, we define two classes of features: objective and subjective. *Objective* features are those that the annotators quickly agree on. This suggests that a user would be very likely to agree with the annotators on the objective features. Therefore, we can conclude that objective features potentially influence a user’s decision of visiting a location. As in the previous example, supposedly, everyone would consider going to a nightlife spot with family is *not* appropriate. Thus a user who regularly goes to nightlife spots might change her mind when they are traveling with her family. As we saw in this example, such *objective* features can directly change users’ decisions adding contextual constraints to the model. *Subjective* features, in contrast, have less impact as they mainly depend on the user’s opinion and personal preferences. If the annotators did not agree on a feature, we would not be able to predict a user’s opinion. Therefore, we cannot predict the influence of subjective features on a user’s decision.

We determined the level of subjectivity or objectivity of features via a crowdsourcing task. In the task, we asked the workers to judge if a location category is appropriate for a context descriptor (e.g., $cat = \text{nightlife-spot}$ and $cxt = \text{with-family}$). We asked at least five different assessors to judge each category-context pair. In the context of this thesis, we define those pairs with high agreement rate between the workers as *objective*, while we consider those lacking assessors agreement as *subjective*. More details on how we created the dataset can be found in Section 3.5.

Table 3.2 lists some example features from our dataset. As we can see in this table, lower values for $|F_{\text{app}}(cat, cxt)|$ mean that the features are more subjective. We created the contextual features for all pairs of 11 contextual dimensions and the 177 most frequent categories of Foursquare category tree². Overall we generated 1,947 contextual features from 11,487 judgments³. More details can be found in Section 3.5.

3.3.2 Training the Classifier

As described earlier, we formulate contextual appropriateness as binary classification. In Section 3.3.1 we explained how we created the contextual features. Here, we describe another dataset for training the appropriateness classifier using our features. We randomly selected 10% of the data from TREC-CS 2016 dataset. We created another crowdsourcing task for annotating the training data. We asked workers to assess if it is appropriate that a user with a full description of context (e.g., Holiday, Friends, Weekend) to visit a location category (e.g., Bar). Each instance in the dataset is considered as appropriate only if at least two of the three assessors voted for their appropriateness. We train the contextual appropriateness classifier on 10% of the data from TREC-CS 2016 to predict the remaining 90% of TREC-CS 2016 and the whole TREC-CS 2015 dataset. We applied a wide range of classifiers for this task. However, we only report the best results that were obtained with SVM [70]. The predicted contextual relevance score is denoted as S_{cxt} and we use it to re-rank the personalized location ranking (see Section 3.4).

²<https://developer.foursquare.com/categorytree>

³The dataset is freely available on request.

3.4 Recommendation based on Information from Multiple LBSNs

After explaining the two major components of our proposed approach in Sections 3.2 and 3.3, here, we explain our way of performing POI recommendation, exploiting the scores from multiple LBSNs. We describe two sets of scores: the frequency-based and review-based scores. We use the frequency-based score to incorporate the boosted keywords (Section 3.2.3), the predicted user tags (Section 3.2.4), as well as other types of information. We also demonstrate how we combine different scores to produce the personalized location ranking using learning to rank.

3.4.1 Frequency-based Score

We base the frequency-based scores on the assumption that users prefer the type of locations that they like more frequently and rate them positively⁴. Therefore, we create positive and negative profiles considering the content of locations in the user's check-in history and calculate the normalized frequencies as they appear in her profile. Then we compute a similarity score between the user's profile and a new location. For simplicity, we only explain how to calculate the frequency-based score using location categories. The method can be easily generalized to calculate the score for other types of information.

Let u be a user and $h_u = \{v_1, \dots, v_n\}$ her history of check-ins. Each location has a list of categories $C(v_i) = \{c_1, \dots, c_k\}$. We define the user category profile as follows:

Definition 3.4.1. A **Positive-Category Profile** is the set of all unique categories belonging to locations that user u has previously rated positively. A **Negative-Category Profile** is defined analogously for locations that are rated negatively.

Each category in the positive/negative category profile is assigned with a user-level normalized frequency. We define the user-level normalized frequency for a category as follows:

Definition 3.4.2. A **User-level Normalized Frequency** for an item (e.g., category) in a profile (e.g., positive-category profile) for user u is defined as:

$$cf_u^+(c_i) = \frac{\sum_{v_k \in h_u^+} \sum_{c_j \in C(v_k), c_j = c_i} 1}{\sum_{v_k \in h_u} \sum_{c_j \in C(v_k)} 1},$$

⁴We consider reviews with rating [4, 5] as positive, 3 as neutral, and [1, 2] as negative.

where h_u^+ is the set of locations that u rated positively. We calculate user-level normalized frequency for negative categories, cf_u^- , analogously.

We create positive/negative category profiles for each user based on Definitions 3.4.1 and 3.4.2. Given a user u and candidate location v , the frequency-based similarity score based on location categories, $S_{cat}(u, v)$, is calculated as follows:

$$S_{cat}(u, v) = \sum_{c_i \in C(v)} cf_u^+(c_i) - cf_u^-(c_i). \quad (3.8)$$

We follow the same procedure to calculate a frequency-based score based on other types of information as listed below:

- S_{key} : We consider location taste keywords (instead of categories) to compute the similarity score between a given user's profile and a candidate location.
- S_{boost} : As for boosting, for each user we follow Definition 3.4.1 to create positive and negative *boosted location taste keyword profiles*. We consider only the location taste keywords that are mapped to user tags (see Section 3.2). Given a candidate location, we calculate boosted keywords similarity score according to Definition 3.4.2 and (3.8).
- S_{ml} : Following Definitions 3.4.1 and 3.4.2, we create positive and negative *user tag profiles* for each user. However, since a candidate location does not naturally have user-assigned tags, we use our ML-based approach to predict user tags. The predicted user tags are then compared with the user's profile following (3.8) to calculate S_{ml} similarity score.
- S_{crf} : We calculate S_{crf} score similar to S_{ml} . We predict user tags for a candidate location using the trained CRF model. Then we follow (3.8) to calculate the S_{crf} score comparing predicted user tags with user profile.
- S_{svm} : This score is also calculated like S_{ml} . For a candidate location, we predict user tags using our trained SVM-based tagging model. The predicted user tags are then compared with the user's profile using (3.8) resulting in S_{svm} .

3.4.2 Review-Based Score

Modeling a user only on locations' content is general and does not determine why the user enjoyed or disliked a POI. The content of locations is often used to

infer “which type” of POIs a user likes. On the other hand, reviews express the motivations behind users’ ratings [199]. Since there could be a lack of explicit reviews from the user, we tackle this sparsity problem using reviews of other users who gave a similar rating to the location. We follow the same idea of Yang et al. [199], that is, a user’s opinion regarding a location could be learned based on the opinions of other users who rated the same location similarly.

We calculate the review-based score using a binary classifier. We model this problem as binary classification since a user, before visiting a new city or location, could get a positive or negative impression of the location after reading the online reviews of other users. We assume that a user compares the characteristics of a location and the opinions which are expressed by other users in their reviews to her expectations and interests. A user would be convinced of visiting a particular location if the reviews satisfy her expectations up to a certain point. An alternative to binary classification would be a regression model. However, we assume that users behave similarly to a binary classifier when they read online reviews before deciding on whether to visit a venue or not. For example, assume a user reads a few positive and negative online reviews about a POI and measures how similar the mentioned qualities are to her expectations. Finally, depending on the balance between the positive remarks and the negative ones, they make a binary decision (i.e., whether to go or not). We see this behavioral pattern similar to that of a binary classifier: it learns from the positive and negative samples and compares the learned parameters with a test sample and assigns its label accordingly. Furthermore, due to data sparsity, grouping ratings as positive and negative aids us to model users more effectively.

We train binary classifier using the reviews from the locations in a user’s check-in history. The positive training samples for user u are positive reviews of locations that were liked by u . Likewise, the negative reviews of locations that u disliked constitute the negative training samples. We decided to ignore the negative reviews of liked locations and positive reviews of disliked locations since they are not supposed to contain any useful information.

We consider TF-IDF score of terms in reviews as features. We trained many classifiers but SVM outperformed all other models. Therefore, we choose SVM and consider the value of its decision function as the review-based score and refer to it as S_{rev} . The decision function gives us an idea on how relevant a location is to a user profile.

Table 3.3. Four proposed models using different combination of similarity scores.

	Category	Review	Keywords	Context
UT-ML	S_{cat}	S_{rev}	S_{key}, S_{ml}	S_{cxt}
UT-CRF	S_{cat}	S_{rev}	S_{key}, S_{crf}	S_{cxt}
UT-SVM	S_{cat}	S_{rev}	S_{key}, S_{svm}	S_{cxt}
PK-Boosting	S_{cat}	S_{rev}	S_{key}, S_{boost}	S_{cxt}

3.4.3 Location Ranking

After defining the mentioned relevance scores, here we explain how we combine them. Given a user and a list of candidate locations, we calculate the scores for each location and combine them to create a ranked list of locations. We adopt several learning to rank⁵ techniques to rank the candidate locations since they have proven to be effective for similar tasks [128, 199]. In particular, we examine the following learning to rank techniques: AdaRank, CoordinateAscent, RankBoost, MART, LambdaMART, RandomForest, RankNet, and ListNet. We introduce four models using different combinations of the scores as mentioned in Table 3.3.

3.5 Data Collection and Analysis

We chose Foursquare and Yelp not only because they are two of the most popular LBSNs where many users leave their check-in data, but also because the type of information provided by Yelp is a perfect complement to the type of information on Foursquare. Moreover, as we will show in the statistics of the dataset, there is a considerable overlap of venues that have a profile on both LBSNs. However, there are places in the TREC dataset that appear only on one of the two crawled LBSNs, hence crawling data from both of them allows making the data gathering more complete.

Deveaud et al. [78] showed that venue-centric features which were extracted from Foursquare play a key role in venue recommendation. On the other hand, Chen et al. [54] argued that user reviews on venues provide a wealth of information that can be leveraged to address the data-sparsity and the cold-start problems

⁵We use RankLib implementation of learning to rank: <https://sourceforge.net/p/lemur/wiki/RankLib/>

for venue recommendation. Also, Yang et al. [199] showed that the accuracy of a recommender system can be significantly improved by extracting opinions from user reviews in Yelp. Also, almost all of the best performing systems in the TREC Contextual Suggestion track 2015 and 2016 [96] crawled data from these LBSNs. In particular, our previous work which were among the best runs in both 2015 [11] and 2016 [4] benefited from a comprehensive crawled dataset from Yelp and Foursquare. We also showed that a system can benefit from multiple LBSNs, and systems using reviews from Yelp and venue tags from Foursquare had the best performance.

3.5.1 Data Crawling

For our collection, we crawled data from Foursquare and Yelp. Foursquare provides an easy-to-access API⁶ which makes crawling quite easy. We used Foursquare's API to crawl a large number of venues and scraped a very smaller fraction of venues for additional information on the website. Yelp's API⁷, on the other hand, has more restrictions and therefore we were able to crawl much fewer venues on Yelp. For TREC 2016 Phase 1,⁸ we only crawled data using the Foursquare's API since there were virtually 630K venues to crawl in a very limited time and thus the only option was using the Foursquare's API. For Phase 2 of TREC 2015 and 2016, there was more time and much fewer venues to crawl so we could crawl data from both LBSNs.

The data was crawled in two time periods: July - August 2015 and July - August 2016. To find the corresponding profiles of venues on LBSNs, we used two search engines: 1) Foursquare venue search engine and 2) Google Custom Search. For TREC 2015 there were 8,794 venues from which we crawled 6,427 places from Yelp and 5,639 from Foursquare, with a considerable overlap between data crawled from Yelp and Foursquare. For TREC 2016 there were 18,808 venues from which we crawled 13,868 places from Yelp and 13,417 from Foursquare, again emerging a big overlap between the two sources. As all the venues are in the US cities, we expected that most of the users who reviewed the venues were English speakers.

Query structure. For each venue in the collection, we created a query to search on the LBSNs. The query consisted of a venue's name and its location. We

⁶<https://developer.foursquare.com/docs/>

⁷<https://www.yelp.com/developers/documentation/v2/overview>

⁸Phase 1 consisted of a live experiment where the participants' runs were evaluated by the TREC assessors.

cleaned the venues' names in the TREC collection since many contained unrelated terms such as the host service (e.g., Facebook, Wikipedia). Finally, the query we used to search for venues was in the form:

query = venue's name + venue's city + venue's state .

Search result validation. Since we could not trust the results of search and in order to minimize the noise, we validated the returned results from the search engine following these steps:

1. We first verified the city and state of the returned venue with the reference data available in the TREC-CS collection.
2. We then measured the similarity between the name of the venue and the name of the returned place using Levenshtein distance.
3. If the similarity between the two names (calculated in Step 2) was more than a threshold (70%), we considered that result as a match. If not, we continued steps 1-3 for other returned results up to the 5th result.

Note that the high similarity threshold (70%) was set to prevent adding possible noise to the collection.

3.5.2 Crowdsourcing

We used the CrowdFlower⁹ crowdsourcing platform to collect judgments of contextual appropriateness of venues and create the additional contextual-appropriateness dataset. We asked a number of crowdworkers to judge if a venue category is appropriate for a trip description. For instance, if a trip was described in the collection as trip type: *business*, trip duration: *one day*, and group type: *family*, for a venue with category *Pizza Place*, then we asked crowdworkers to judge if the venue was appropriate to the trip. In particular, we asked them: “Is a *Pizza Place* appropriate for a *business* trip?”, “Is it appropriate to go to a *Pizza Place* on a *one-day* trip?”, “Is it appropriate to go to a *Pizza Place* with *family*?” While assessing such tasks could seem trivial and objective, it is in fact subjective in many cases (e.g., going to a *pharmacy* with *family*). Therefore, we asked at least 5 crowdworkers to provide their judgment to each row. If we found no agreement among the assessors, we considered the task as subjective. We considered the answer “appropriate” as a +1 score and “inappropriate” as a -1 score. Thus,

⁹<http://www.crowdfLOWER.com>

the assessment agreement is the average of assessment scores. We asked workers to judge the context/category pairs for almost all possible pairs regardless of their existence in the TREC collections. This makes this dataset general enough to be used for other purposes.

We made sure to explain the task clearly to the workers and asked them to assess such appropriateness regardless of their personal preferences over categories. Also, we performed a training step and allowed only top-quality workers to do the task.

3.5.3 Data Analysis

The released collection contains more than 330K venues from Foursquare for TREC 2016 Phase 1 and 15,765 venues from both Foursquare and Yelp for TREC 2016 Phase 2.¹⁰ As we can see in Table 3.4 there were many broken or unrelated links in the TREC collection (300K out of 600K), however, there were much fewer unrelated links for Phase 2 (3K out of 18K). For each venue we release all available information: venue name, address, category, tags, ratings, reviews, check-in count, menu, opening hours, parking availability, etc.

The contextual-appropriateness collection consists of 1,969 pairs of trip descriptors and venue categories as features. In order to enable researchers to train their models using the contextual appropriateness of venues, we created another collection providing ground truth assessments for the contextual appropriateness of the venue categories. It contains the contextual information (i.e., trip type, group type, trip duration) for 10% of the whole TREC collection. This collection contains 760 rows including the features we already created using crowdsourcing and the context-appropriateness labels for venues. The 10% of labeled data allows to model the venues' contextual appropriateness given the users' context and to make prediction for the remaining 90% of the data, as we did in [4].

Tables 3.4 and 3.5 lists some statistics of the crawled collection and the crowdsourced collection, respectively. Figure 3.4 shows the histogram of venue-appropriateness features assessed by the workers. We divided the assessments in three groups based on appropriateness scores: $[-1.00, -0.40]$: not appropriate (objective), $[-0.40, +0.40]$: no agreement (subjective), $(+0.40, +1.00]$: appropriate (objective). To categorize the tasks as subjective and objective, we assumed that those tasks for which there was a high agreement between the assessors could be considered objective since everybody agreed on their (in)appropriateness. On the other hand, we assumed that those tasks with relatively lower agreement

¹⁰Phase 2 was a reranking task where an offline evaluation was performed.

Table 3.4. Statistics on the crawled collection

	Phase 1	Phase 2	
	TREC-CS 16	TREC-CS 15	TREC-CS 16
# requests	442	211	442
# requests evaluated by TREC	58	211	58
# venues in TREC collection	633,009	8,794	18,808
# venues crawled: Yelp	-	6,427	13,868
# venues crawled: Foursquare	336,080	5,639	13,417
# Yelp and Foursquare overlap	-	4,844	11,520
Avg. reviews per venue	-	117.34	66.82
Avg. categories per venue	1.35	1.63	1.57
Avg. taste keywords per venue	-	8.73	7.89
Avg. user tags per user	3.61	1.46	3.61
# distinct user tags	150	186	150

Table 3.5. Statistics on the crowdsourced contextual appropriateness collection

# categories	179
# category-context pairs	1969
# assessments	11,487
Avg. assessments per pair	5.83
Avg. assessment agreement	85%
# full travel annotations	760

between the assessors could be considered subjective.

3.6 Experimental Setup

In this section, we present the experimental settings including the datasets we used, compared methods, and evaluation process.

3.6.1 Data

Recommendation effectiveness. We evaluate our approach on two benchmark collections, published by TREC. The collections are those used in the *Batch Ex-*

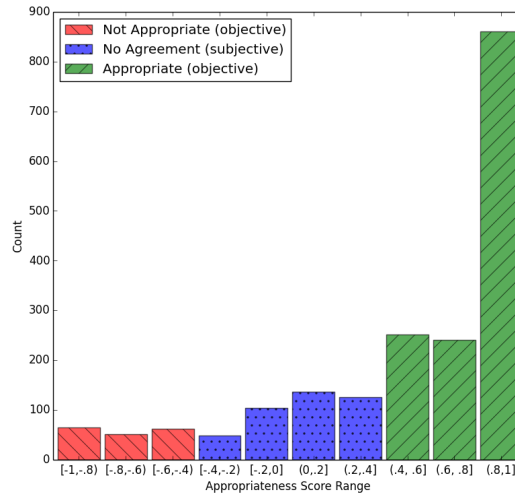


Figure 3.4. Histogram of venue-context appropriateness score ranges. We partition the histogram into 3 parts based on the scores range. Scores below -0.4 represent inappropriateness and score higher than $+0.4$ represent appropriateness. Scores between -0.4 and $+0.4$ do not provide much information and show no agreement among assessors (subjective task).

periments/Phase 2 of the TREC-CS track 2015 [76] and 2016 [96]. The task was to rank a list of candidate locations in a new city for a user, given her history of check-ins in other cities. The datasets were collected using crowdsourcing where each user rated 30 to 60 locations in one or two cities. In addition, each user may have tagged locations to explain why they like them (i.e., user tags). Later, the same users were called to rate new POIs in another city as well as the contextual factors of their trip. More details can be found in Section 3.5.

Dimensionality reduction. We evaluate the dimensionality reduction effectiveness on the same datasets as we do for recommendation effectiveness. We compare the performance of our proposed model to a well-known dimensionality reduction model (i.e., PCA) in terms of recommendation effectiveness. Therefore, we use the same datasets used to evaluate recommendation effectiveness, however, we provide more details and discussion related to dimensionality reduction.

User tag prediction. Since the test set in TREC-CS 2015 and TREC-CS 2016 do not include user tags for locations, we need to evaluate the user tag prediction on the training datasets. Therefore, we randomly split the TREC-CS 2015 and TREC-CS 2016 training sets into: training, development, and test set. We train the

Table 3.6. Statistical details of user tagging dataset

	Training Set	Test Set
# instances	20,148	5,037
# non-null tags	102,667	25,541
# null tags	54,444	13,954
# unique user tags	156	121
# unique location keywords	2,676	1,398
Avg. user tags per location	1.85	1.82
Avg. keywords per location	5.10	5.07

taggers using the new training set, tune them using the new development set and evaluate them with the new test set. As part of the evaluating recommendation effectiveness we show how different taggers can improve the recommendation; however, the aim of this experiment is to show how accurately we can model the user interests and tagging behavior. The statistical details of the tags dataset is listed in Table 3.6.

3.6.2 Metrics

We evaluate the recommendation effectiveness as well as the dimensionality reduction for the top- k recommendation. We also evaluate the effectiveness of user tag prediction methods.

Recommendation effectiveness. In both TREC-CS 2015 and TREC-CS 2016 datasets, for each user u , the data, $S(u)$, is split into two sets: a number of locations visited in one or two cities constitute the training set and a number of locations in a new city constitute the test set. Given a user u , if a recommended location in the test set is marked by the user as *relevant*, it is a “hit,” otherwise it is a “miss.” To perform a fair comparison, we choose the official evaluation protocol and metrics of TREC-CS for this task, which are P@5 (Precision at 5), nDCG@5 (Normalized Discounted Cumulative Gain at 5), and MRR (Mean Reciprocal Rank). Since the main focus in this task was to improve the location rankings, such evaluation metrics serve as perfect metrics.

The relevance assessments for test sets are slightly different in TREC-CS 2015 and TREC-CS 2016. In TREC-CS 2015 relevance of a location to a user is defined with a binary value with 0 as irrelevant and 1 as relevant, whereas in TREC-CS 2016 users rated locations in the range of -2 to $+2$. This also explains why

the main evaluation metric in TREC-CS 2015 is P@5 as opposed to nDCG@5 in TREC-CS 2016. Both P@k and nDCG@k metrics are evaluated over k top locations on the ranked list. Let U be the set of users and r_u^p be the rating score assigned by user u to the location at the i^{th} rank of the list. Precision and nDCG values are calculated at the k^{th} position as follows:

$$P_u@k = \frac{\#hits_u@k}{k},$$

$$nDCG_u@k = Z_u \sum_{i=1}^k \frac{2^{r_u^i} - 1}{\log(1 + i)},$$

where u is the given user, Z_u is a normalization factor and $\#hits_u@k$ is the number of relevant locations for user u in the top- k locations of the ranked list. nDCG@k and P@k are the mean of $nDCG_u@k$ and $P_u@k$ over U respectively. MRR is also calculated as follows:

$$MRR = \frac{1}{|U|} \sum_{u=1}^{|U|} \frac{1}{rank_u},$$

where $rank_u$ is the ranking of the first relevant location for user u . We conduct a 5-fold cross validation on the training data to tune our model. We determine the statistically significant differences using the two-tailed paired t-test at a 95% confidence interval ($p < 0.05$).

Dimensionality reduction. Since there is no ground truth data to evaluate dimensionality reduction methods, we evaluate the recommendation effectiveness using different dimensionality reduction methods to see how they enhance the overall recommendation. Therefore, we use the same evaluation metrics that we used for evaluating recommendation effectiveness.

User tag prediction. Since we modeled the user tag prediction problem as a sequence-labeling problem, we evaluate the effectiveness of user tag prediction using the same metrics used for evaluating typical sequence-labeling problems such as *Part of Speech* (POS) tagging. Therefore, we report Precision, Recall, and F-Measure for this experiment. Let T_p the number of true positive and F_p the number false positive non-null predicted tags. Then precision is calculated as follows:

$$Precision = \frac{T_p}{T_p + F_p}.$$

Given the number of false negatives, F_n , we also calculated recall as follows:

$$Recall = \frac{T_p}{T_p + F_n}.$$

F-measure is then defined as follows:

$$F - Measure = 2 \times \frac{Precision \times Recall}{Precision + Recall}.$$

3.6.3 Compared Methods

Recommendation effectiveness. We consider the best performing system in TREC-CS 2015 as our baseline. Moreover, we compare our proposed method with state-of-the-art context-aware POI recommendation methods. We also compare our proposed PK-Boosting with other models based on user tag prediction (i.e., UT-ML, UT-CRF, and UT-SVM).

- **LinearCatRev** is our previous work [11] which is the best performing model of TREC-CS 2015. It extracts information from different LBSNs and uses it to calculate category-based and review-based scores. Then, it combines the scores using linear interpolation. We choose this baseline for two reasons, firstly because it is the best performing system of TREC-CS 2015, and secondly because it also uses scores derived from different LBSNs.
- **GeoSoCa** exploits geographical, social, and categorical correlations for POI recommendation [216]. GeoSoCa models the geographical correlation using a kernel estimation method with an adaptive bandwidth determining a personalized check-in distribution. It models the categorical correlation by applying the bias of a user on a POI category to weigh the popularity of a POI in the corresponding category modeling the weighted popularity as a power-law distribution. We used the implementation of GeoSoCa released in [130]. We did not include the social correlation component since such information does not exist in the datasets.
- **n-Dimensional Tensor Factorization (nDTF)** [108] generalizes matrix factorization to allow for integrating multiple contextual features into the model. We used the publicly available implementation of nDTF¹¹. Regarding the features, we include two types of features: (1) location based: category, keywords, average rating on Yelp, and number of ratings on Yelp (as an indicator of its popularity); (2) user based: age group and gender.
- **UT-ML** differs from PK-Boosting in one score. For UT-ML, instead of the keyword boosting score (S_{boost}), we use the score based on the predicted

¹¹<https://github.com/VincentLiu3/TF>

user tags following maximum likelihood criterion. As we described in Section 3.2.4, we also explored three different models as alternatives to PK-Boosting. Our aim is to study the impact of predicting user tags on recommendation effectiveness, compared to PK-Boosting. The other two alternative approaches are listed as follows.

- **UT-CRF** predicts user tags using a trained CRF model. Then, for each venue-user pair, it computes the similarity between the predicted user tags and the user profile. Finally, it replaces the boosting score with the computed similarity score (see Section 3.2.4). We used CRFSuite¹² implementation of CRF.
- **UT-SVM** predicts user tags given a user-venue pair using an SVM-based tagging model. The boosting score is replaced by the similarity score between the user profile and predicted user tags (see Section 3.2.4). We used YamCha¹³ implementation of the SVM-based tagging model.

Dimensionality reduction. In order to evaluate the keyword boosting approach from the perspective of dimensionality reduction, we also apply the following well-known dimensionality reduction method to reduce the location keywords dimension. In particular, we use *PK-PCA*. PK-PCA uses Principal Component Analysis (PCA) to reduce the dimensionality of location keywords. Finally, we consider the score computed based on the PCA model as the boosting score and calculate the recommendation score accordingly.

User tag prediction. As user tags contain very crucial information explicitly described by users, we aim to evaluate the effectiveness of user tag models. In particular, we evaluate the following models:

- *Conditional Random Fields (CRF) Tagger* [119] models the sequence tagging problem in a discriminative manner. The tagger is based on binary features that are extracted from the text and optimized for the training data.
- *SVM-based Tagger* [117] is also a discriminative approach that trains one SVM classifier per tag. The model is an ensemble of all SVM classifiers.

¹²<http://www.chokkan.org/software/crfsuite/>

¹³<http://chasen.org/~taku/software/yamcha/>

3.7 Results and Discussion

In this section, we first present the results for recommendation effectiveness. We also show the results for location keyword dimensionality reduction and user tag prediction. Furthermore, we study the effect of different sources and scores on recommendation effectiveness.

3.7.1 Performance Comparison

Tables 3.7 and 3.8 demonstrate the performance of our approach compared with other methods for the TREC-CS 2015 and TREC-CS 2016 datasets, respectively. We choose the best performing learning to rank technique for each model we adopt the best performing learning to rank technique according to Tables 3.9 and 3.10. It is worth noting that the best learning to rank technique for PK-Boosting is ListNet [51]. Tables 3.7 and 3.8 show that PK-Boosting outperforms the competitors in terms of the three evaluation metrics. This indicates that the proposed PK-Boosting approach improves the performance of POI recommendation. This happens because the proposed approach for boosting location keywords addresses the data sparsity problem, while at the same time it captures user preferences more accurately. In contrast, the models UT-ML, UT-CRF, and UT-SVM introduce a prediction error, when predicting user tags for a candidate location. This error is then propagated to location ranking and subsequently degrades the models' performances. As we can see, GeoSoCa and nDTF exhibit the worst performance among all compared methods. This happens mainly because these methods rely on user-POI check-in associations among the training and test sets. In other words, there should be enough common POIs appearing in both the training and test sets, otherwise they fail to recommend unseen POIs. Hence, they suffer from the high level of sparsity on these datasets. In particular, the intersection of POIs in the training and test sets is 771 (out of 8,794) and 4 (out of 18,808) in TREC-CS 2015 and 2016, respectively.

To compute the review-based classifier, we used various classifiers such as Naïve Bayes and k-NN; however, the SVM classifier exhibited a better performance by a large margin. The SVM classifier is a better fit for this problem since it is more suitable for text classification, which is a linear problem with weighted high dimensional feature vectors. Also, we observed a significant difference between the number of positive reviews and negative reviews per location. Generally, locations receive more positive reviews than negative reviews and, in our case, this results in a unbalanced training set. Most of the classification algorithms fail to deal with the problem of unbalanced data. This is mainly due to

the fact that those classifiers try to minimize an overall error rate. Therefore, given an unbalanced training set, the classifier is usually trained in favor of the dominant class to minimize the overall error rate. However, SVM does not suffer from this, since it does not try to directly minimize the error rate but instead tries to separate the two classes using a hyperplane maximizing the margin. This makes SVM more intolerant of the relative size of each class. Another advantage of linear SVM is that the execution time is very low and there are very few parameters to tune.

3.7.2 Impact of Different Learning to Rank Techniques

In this experiment we aim to show how the recommendation effectiveness is affected by applying different learning to rank techniques to combine the scores. Our aim is to show how different state-of-the-art learning to rank models can effectively combine the proposed scores for recommendation. Tables 3.9 and 3.10 report P@5 applying different learning to rank techniques for TREC-CS 2015 and TREC-CS 2016 respectively. We report the performance for UT-ML, UT-CRF, UT-SVM, and PK-Boosting. As we can see, ListNet in many cases outperforms other learning to rank techniques. More specifically, for TREC-CS 2015, ListNet exhibits the best performance for all models except for UT-ML. It is very interesting that RankNet exhibits the best performance for UT-ML, and both ListNet and RankNet are based on artificial neural networks. As for TREC-CS 2016, RankNet performs better for UT-ML and UT-SVM while ListNet performs better for other models. As we can observe, applying different learning to rank techniques can potentially have a big impact on recommendation results. Therefore, it is critical to apply the best technique for the scores.

3.7.3 Impact of Using Information from Multiple LBSNs

Tables 3.11 and 3.12 evaluate the performance of the examined models before and after removing information from each LBSN. In this set of experiments, we also report the relative performance drop of different models when using information from the two different LBSNs. As we can see in almost all cases, when a source of information is removed from the model, we observe a drop in the performance. The average drop for TREC-CS 2015 is -4.90% and for TREC-CS 2016 is -6.00% which confirms the effectiveness of exploiting information from different LBSNs. This indicates that using multimodal information from different LBSNs is a key to improve POI recommendation. For all different runs, the

Table 3.7. Performance evaluation on TREC-CS 2015.

	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	0.5858	-	0.6055	-	0.7404	-
GeoSoCa	0.5147*	-12.14	0.5404*	-10.75	0.6918*	-6.56
nDTF	0.5232*	-10.96	0.5351*	-11.63	0.6707*	-9.41
UT-ML	0.6224*	6.25	0.6320*	4.38	0.7496	1.24
UT-CRF	0.6249*	6.67	0.6285	3.80	0.7434	0.41
UT-SVM	0.6219*	6.16	0.6339*	4.69	0.7553	2.01
PK-Boosting	0.6259*	6.85	0.6409*	5.85	0.7704*	4.05

Bold values denote the best scores and the superscript * denotes significant differences compared to LinearCatRev. Δ values (%) express the relative improvement, compared to LinearCatRev. For each model we report the scores using the best learning to rank technique (Table 3.9).

Table 3.8. Performance evaluation on TREC-CS 2016.

	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	0.4897	-	0.3213	-	0.6284	-
GeoSoCa	0.4207*	-14.09	0.2958	-7.94	0.6497	3.39
nDTF	0.4172*	-14.80	0.2663*	-17.12	0.6167	-1.86
UT-ML	0.5138	4.92	0.3357	4.48	0.6389	1.67
UT-CRF	0.5138	4.92	0.3410	6.13	0.6765	7.65
UT-SVM	0.5207	6.33	0.3389	5.48	0.6510	3.60
PK-Boosting	0.5310	8.43	0.3526*	9.74	0.6800	8.21

Bold values denote the best scores and the superscript * denotes significant differences compared to LinearCatRev. For each model we report the scores using the best learning to rank technique (Table 3.10).

Table 3.9. Effect on P@5 for different learning to rank techniques in TREC-CS 2015.

	UT-ML	UT-CRF	UT-SVM	PK-Boosting
MART	0.5911	0.6008	0.5958	0.6010
RankNet	0.6224	0.6190	0.6155	0.6190
RankBoost	0.6030	0.6086	0.6088	0.6146
AdaRank	0.6028	0.6121	0.6117	0.5893
CoordinateAscent	.06115	0.5858	0.5918	0.5997
LambdaMART	0.6022	0.6077	0.6061	0.6135
ListNet	0.6069	0.6249	0.6219	0.6259
RandomForests	0.5836	0.5966	0.5920	0.5963

Bold values denote the best learning to rank technique per model.

Table 3.10. Effect on P@5 for different learning to rank techniques in TREC-CS 2016.

	UT-ML	UT-CRF	UT-SVM	PK-Boosting
MART	0.4653	0.4103	0.3931	0.4483
RankNet	0.5138	0.5103	0.5237	0.5103
RankBoost	0.3414	0.4241	0.4345	0.4586
AdaRank	0.3414	0.3414	0.3414	0.3414
CoordinateAscent	0.5021	0.4931	0.4931	0.5000
LambdaMART	0.3793	0.3931	0.3793	0.4931
ListNet	0.5103	0.5138	0.5103	0.5310
RandomForests	0.4207	0.4069	0.4345	0.4310

Bold values denote the best learning to rank technique per model.

Table 3.11. Performance evaluation after removing information provided by Four-square (F) and Yelp (Y) in the TREC-CS 2015 dataset.

	F	Y	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	✓	✓	0.5858	-	0.6055	-	0.7404	-
	✓	✗	0.5649	-3.57	0.5860	-3.22	0.7263	-1.90
	✗	✓	0.5697	-2.75	0.5917	-2.28	0.7341	-0.85
UT-ML	✓	✓	0.6224	-	0.6320	-	0.7496	-
	✓	✗	0.5288*	-15.04	0.5307*	-16.03	0.6487*	-13.46
	✗	✓	0.5787*	-7.02	0.5746*	-9.08	0.6833*	-8.84
UT-CRF	✓	✓	0.6249	-	0.6285	-	0.7434	-
	✓	✗	0.5960*	-8.95	0.5930*	-5.65	0.7301	-1.79
	✗	✓	0.6055	-3.10	0.6238	-0.75	0.7503	0.93
UT-SVM	✓	✓	0.6219	-	0.6339	-	0.7553	-
	✓	✗	0.5728*	-7.90	0.5921*	-6.59	0.7388	-2.18
	✗	✓	0.6129	-1.45	0.6250	-1.40	0.7497	-0.74
PK-Boosting	✓	✓	0.6259	-	0.6409	-	0.7704	-
	✓	✗	0.5731*	-8.44	0.6010*	-6.23	0.7602	-1.32
	✗	✓	0.6044	-3.44	0.6227	-2.84	0.7613	-1.18

The superscript * denotes significant differences compared to the performance each model has when using information from the two different LBSNs. Δ values (%) express the relative drop, compared to the performance each model has when using information from the two different LBSNs. (Average drop = -4.90%)

best performing method is the proposed PK-Boosting, that uses a combination of information from both LBSNs.

3.7.4 Impact of Using Different Scores

In this experiment, we try to demonstrate the effectiveness of each score. We remove each score and analyze our model’s performance without it (but we do not remove more than one score at a time). The results are reported in Table 3.13. The first line (*All*) shows the results for P@5, nDCG@5, and MRR using all scores. The second line ($-S_{cat}$) shows the results without the location categories, and so on for the other lines.

The results show a decrease of the model’s performance after removing each of the scores exhibiting an average relative drop of -4.31%. It indicates that our system is able to capture different aspects of information and combine them to

Table 3.12. Performance evaluation after removing information provided by Four-square (F) and Yelp (Y) in the TREC-CS 2016 dataset.

	F	Y	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	✓	✓	0.4897	-	0.3213	-	0.6284	-
	✓	✗	0.4172*	-14.49	0.2705*	-15.81	0.6222	-0.99
	✗	✓	0.4759	-2.46	0.3072	-4.39	0.6032	-4.01
UT-ML	✓	✓	0.5138	-	0.3357	-	0.6389	-
	✓	✗	0.4862	-5.37	0.3079	-8.28	0.6038	-5.49
	✗	✓	0.5034	-2.02	0.3313	-1.31	0.6393	0.06
UT-CRF	✓	✓	0.5138	-	0.3410	-	0.6765	-
	✓	✗	0.5069	-1.34	0.3336	-2.17	0.6531	-3.46
	✗	✓	0.4793	-6.71	0.3133	-8.12	0.6268	-7.35
UT-SVM	✓	✓	0.5207	-	0.3389	-	0.6510	-
	✓	✗	0.4724	-9.28	0.3057*	-9.80	0.6260	-3.84
	✗	✓	0.4793	-7.95	0.3158	-6.82	0.6512	0.03
PK-Boosting	✓	✓	0.5310	-	0.3526	-	0.6800	-
	✓	✗	0.4793*	-9.74	0.3210	-8.39	0.6542	-3.79
	✗	✓	0.4759*	-10.38	0.3177*	-9.90	0.6354	-6.56

The superscript * denotes significant differences compared to the performance each model has when using information from the two different LBSNs. (Average drop= -6.00%)

Table 3.13. Performance of PK-Boosting using all the scores (All) and after removing each score at a time.

	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
<i>All</i>	0.6259	-	0.6409	-	0.7704	-
$-S_{cat}$	0.6009*	-3.99	0.6124*	-4.45	0.7324*	-4.93
$-S_{rev}$	0.5555*	-11.25	0.5837*	-8.92	0.7383*	-4.17
$-S_{key}$	0.6009*	-3.99	0.6113	-3.35	0.7443	-3.10
$-S_{boost}$	0.6190	-1.10	0.6312	-1.51	0.7610	-1.22
$-S_{cxt}$	0.5962*	-4.75	0.6126*	-4.42	0.7437	-3.47

The superscript * denotes significant differences compared to the model using all scores (*All*). Percentages in bold represent the highest decrease in performance when the corresponding score is removed (Average relative drop = -4.31%).

create a better personalized ranking model for POI recommendation. The S_{cat} score models the types of locations a user is interested in visiting, while S_{rev} models the reasons the user likes/dislikes different locations belonging to the same category. S_{key} tries to incorporate the most important keywords extracted from the reviews and to describe a location and its characteristics. S_{boost} boosts the most important keywords that interest a user and the contextual relevance is measured by S_{cxt} . Our model exhibits its largest decrease in performance when S_{rev} is removed from the model. This suggests that the review-based score is the most important score in our model. We think this is because it captures users' opinions. In fact, it is crucial to realize why a user rates two locations in the same category differently.

3.7.5 Impact of Number of Visited POIs

We report P@5 of all models on TREC-CS 2015 and TREC-CS 2016 in Figure 3.5. In this set of experiments, we vary the number of locations to find the mapping between the taste keywords and the user tags. We calculate the scores of Section 3.4 with different number of locations and train the ranking model. Figure 3.5 shows that PK-Boosting is the winning method when compared with other models for all different number of locations. This result indicates that PK-Boosting is more robust when the training set is smaller, whereas the prediction models ML and SVM are not very well trained using such a small data and their performance gets worse.

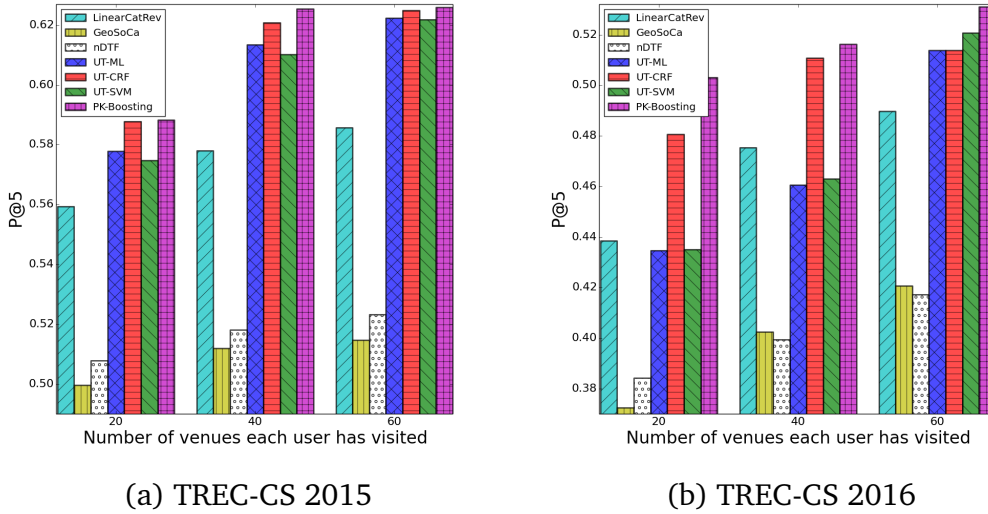


Figure 3.5. Effect on P@5 by varying the number of locations that each user has visited for (a) TREC-CS 2015 and (b) TREC-CS 2016.

3.7.6 Impact of Visiting POIs from a Single City vs. Two Cities

In this experiment, we intend to see how the number of visited locations from one single city affects the performance of our model as opposed to the same number of locations from different cities. In order to do that, we consider at maximum 2 cities, and we train our model using 10, 15, ..., 60 of them for each user as their history of preferences. To make sure that the order of selected locations does not affect our experiment, we shuffle the list of previously visited locations in two ways: 1) we make sure that the first 30 locations are from a city and the second 30 are from the other city; 2) we shuffle the order of visited locations for each city and interleave them. For example, v_1 would be a location from City1, v_2 a location from City2, v_3 a location from City1, and so on. We conduct this experiment with 5 differently shuffled lists and report the average of the results.

The first ordering method ensures that the first half of the locations, visited by a user, are from a particular city. The second ordering, on the other hand, makes sure that for a given number of visited locations n , $n/2$ of them are locations from City1 and $n/2$ are from City2. We intend to examine how our model performs when we have information about users from only one city as opposed to two cities. Moreover, in cases where we have a low number of visited locations in the user's history, it is interesting to see how our model performs in both scenarios (i.e., all locations from a single city vs. from multiple cities).

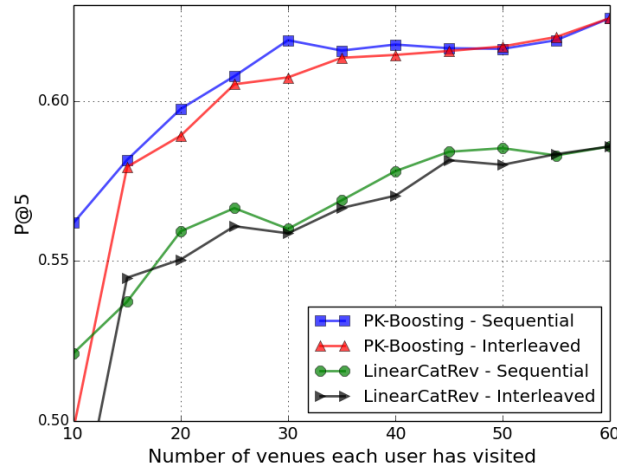


Figure 3.6. Our model’s performance in terms of P@5 with different number of locations as users’ history of preferences compared to LinearCatRev. We have chose the order of locations in two different manners. Sequential: the first 30 locations are from one single city, the second 30 are from another city, Interleaved: the list of locations is interleaved based on their cities.

Figure 3.6 demonstrates our system performance in terms of P@5 with different number of locations in the user’s history compared to LinearCatRev. As we can see in the figure, the model shows a large improvement up to the first 30 locations, decreasing in size after we add 40 locations in both orderings. However, it is interesting to see that the sequential order always performs better than the interleaved one. This difference is more evident when the number of locations is smaller than 20. It suggests that when we have limited number of locations as the user’s history, it is better to have them all about the same city. This can be observed when there are 30 locations and all from one single city (denoted as *Sequential*), we get a much better performance as compared to training the model using 30 locations with half from one city and the other half from another one (denoted as *Interleaved*).

3.7.7 Dimensionality Reduction

In this experiment we compare our personalized keyword boosting method with the well-known dimensionality reduction method PCA. Since keyword boosting is a kind of personalized dimensionality reduction, we choose to compare our method with PCA. As Tables 3.14 and 3.15 show, our personalized keyword

boosting method is able to beat PCA with respect to recommendation effectiveness in terms of both $P@k$ and $nDCG@k$. It suggests that the proposed probabilistic model is able to effectively reduce the dimensionality of location keywords taking into account user personal preferences as well as interests. In fact, in TREC-CS 2015 the average location keyword per user is 277 and our proposed approach is able to reduce it to 41 (−%85), whereas PCA reduces it to 25 (−%91). Moreover, the average location keywords per user in TREC-CS 2016 is 302 and PK-Boosting reduces it to 105 (−%65) compared with PCA reducing it to 16 (−%95).

It is worth noting that PCA produces $\min(n, m - 1)$ principal components, where n is the number of data samples and m is the number of data dimensions. Since in both datasets $n \ll m$, the number of principal components is bounded by n and thus PCA reduces data dimensionality more than PK-Boosting. This difference for TREC-CS 2016 is even bigger due to the fact that users have smaller number of previously visited locations. It is interesting to note the difference between the dimensionality reduction that PK-Boosting exhibits on TREC-CS 2015 and TREC-CS 2016 (i.e., 41 vs. 105). This is due the difference between the average number of user tags in the two datasets. As we can see in Table 3.4, the average user tag per user for TREC-CS 2015 is 1.46 as compared with 3.61 for TREC-CS 2016. Since PK-Boosting personalizes dimensionality reduction problem according to user tags, the average number of user tags can potentially have an impact on the average number of reduced dimensions. The results of Tables 3.14 and 3.15 show that PK-Boosting outperforms PCA in terms of recommendation effectiveness even though PCA is able to reduce more dimensions. This suggests that incorporating personal information for dimensionality reduction is effective and hence PK-Boosting performs better.

3.7.8 User Tag Prediction

In this experiment we evaluate the effectiveness of different user tag prediction methods. The aim of this experiment is to show how effective user tag prediction is in terms of user tag prediction accuracy. In previous experiments we showed how user tag prediction can improve the overall recommendation effectiveness; however, it is crucial to know how effective is the prediction model so that we can further analyze and improve the prediction accuracy in order to achieve better recommendation. Table 3.16 reports the performance of different user tag prediction models. Note that CRF and SVM-based taggers are trained using the same feature set for fair comparison. As we can see in this table the SVM based model is able to beat all other models. In fact, the SVM based model benefits highly

Table 3.14. Performance comparison on TREC-CS 2015 on dimensionality reduction.

	Avg. Dim.	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	277	0.5858	-	0.6055	-	0.7404	-
PK-PCA	25	0.6030*	4.37	0.6157	3.07	0.7366	-0.32
PK-Boosting	41	0.6259*	6.85	0.6409*	5.85	0.7704*	4.05

The superscript * denotes significant differences compared to LinearCatRev.

Table 3.15. Performance comparison on TREC-CS 2016 on dimensionality reduction.

	Avg. Dim.	P@5	$\Delta(\%)$	nDCG@5	$\Delta(\%)$	MRR	$\Delta(\%)$
LinearCatRev	302	0.4897	-	0.3213	-	0.6284	-
PK-PCA	16	0.5106	4.21	0.3406	6.02	0.6424	2.23
PK-Boosting	105	0.5310	8.43	0.3526*	9.74	0.6800	8.21

The superscript * denotes significant differences compared to LinearCatRev.

Table 3.16. Performance comparison of user tag prediction models.

	Precision	Recall	F-Measure
ML	0.3982	0.2421	0.3011
SVM-Based	0.7923	0.8110	0.8016
CRF	0.7646	0.7573	0.7609

from the features that are extracted using the proposed mapping and therefore it can beat ML.

3.8 Summary

In this chapter, we presented a probabilistic model to find the mapping between user tags and location taste keywords. This mapping enabled us to explore various directions to address the data sparsity problem for POI recommendation. In particular, we followed two directions: 1) a PK-Boosting model to reduce the dimensionality of location taste keywords and 2) three models to predict user tags for a new location, as alternatives to PK-Boosting. Moreover, we described how to incorporate the new information into POI recommendation, calculating different scores from information from multiple LBSNs. In addition, we also

created a dataset to measure the contextual appropriateness of locations and explained how we used the dataset to improve our model. Following learning to rank techniques, the final POI recommendation ranking is obtained based on the computed scores. The experimental results on two TREC collections demonstrate that our method outperforms state-of-the-art strategies. This confirms that the proposed approach, PK-Boosting, addresses the data sparsity problem capturing user preferences accurately.

Although the results of this chapter demonstrate the success of our proposed content-based approach, our intuition is that such content-based approaches are able to outperform collaborative approaches in cases where the data is very sparse. Data sparsity is a known problem in recommendation, however, the TREC-CS dataset is an example of very sparse data. This is mainly due to the fact that the training data is collected from a random city in the U.S. and the test data is collected from another city. Therefore, there is not much common check-in records in the test and train datasets. As a consequence, collaborative approaches are not able to learn the latent associations between users and POIs [25]. Therefore, in the next chapter, we propose a collaborative approach to model users and POIs.

Chapter 4

Collaborative User Modeling for Venue Suggestion

4.1 Introduction

While content-based approaches are strong in modeling user, POI behavior and profile, they fail to capture the collaborative behavior of users on LBSNs. This happens because every user is modeled solely based on their past behavior. Although our first approach in Chapter 3 demonstrated the power of content-based approaches in the absence of collaborative data, the available data from the major LBSNs often contains sufficient data for learning user-POI associations. Therefore, in this chapter we present a collaborative approach for POI recommendation. More specifically, we propose a two-phase collaborative ranking (CR) algorithm that is able to learn users' preferences from implicit feedback (i.e., check-ins). Our model is inspired by the successful results of CR in other domains with explicit user feedback [167, 189] and of ranking methods using implicit user feedback for POI recommendation [125, 165]. It is worth noting that many temporal and long-term characteristics of users behavior and POI popularity should be studied on large-scale check-in data from LBSNs. That is why we evaluate our proposed model on two datasets from Foursquare and Gowalla. However, since users often do not leave explicit feedback about their check-ins, we have taken into account users' implicit feedback. Therefore, we use a two-phase implicit feedback inference in our algorithm. In fact, we assume that a single check-in means that a user "likes" the POI while multiple visits mean that user prefers the POI over others. Based on this assumption, in the first phase we push POIs with single or multiple check-ins at the top of the recommendation list, taking into account the geographical influence of POIs in the same neigh-

borhood. In the second phase, we push POIs with multiple check-ins over the ones with a single visit. As argued in [125], considering both visited and unvisited POIs in the learning alleviates the sparsity problem. Therefore, the first phase of our algorithm addresses the sparsity problem, whereas the second stage boosts the accuracy of our model by pushing more relevant POIs at the top of the list. To take into account the dynamics of user and place check-in, we introduce a *time-sensitive regularizer* in the ranking loss. The regularizer models the activity pattern of every user and venue over time. Adding this regularizing parameter to the objective function fuses the activity patterns into the ranking function in a collaborative way.

Our contributions in this chapter can be summarized as follows:

- We perform an extensive analysis to demonstrate the underlying patterns of preference and popularity over time.
- We propose a general time-sensitive regularizer, taking into account the variance of users activities and venues popularity over time.
- We propose a novel two-phase CR-based POI recommendation algorithm incorporating users implicit check-in feedback with a focus on the top of the list.
- We propose a geographical similarity measure and add its influence to the model's objective function.

Experiments on two benchmark datasets show that the proposed approach outperforms state-of-the-art POI recommendation and CR methods. In particular, we show that the joint learning strategy enables the model not only to address the sparsity problem but also to rank relevant POIs higher. The first phase mainly addresses the sparsity problem by adding the geographical influence as well as considering both visited and unvisited venues for training. The second phase improves the accuracy of the model, pushing to the top of the recommendation list the POIs that users prefer more. Moreover, we demonstrate the effectiveness of the time-sensitive regularizer, that is applied to both phases of the algorithm taking into consideration the long-term behavior of users and the popularity of POIs.

The remainder of the chapter is organized as follows, a deep analysis on the datasets is performed in Section 4.2. We describe our collaborative method in Section 4.3 and in Section 4.4, we describe the evaluation settings. Section 4.5 evaluates the performance of our model against competitive models. Finally, Section 4.6 summarizes the chapter, highlighting the need for combining

the content-based and collaborative approaches for improved recommendation, which is described in the next chapter.

4.2 Data Analysis

In this section, we conduct an extensive analysis of two real-world datasets to explore user preferences' dynamics over time.

4.2.1 Data

We use two real-world check-in datasets from Foursquare and Gowalla provided by [208]¹. The Foursquare's dataset originally consists of 342,850 check-ins of users in Singapore in the period of Aug. 2010 to Jul. 2011 [208]. The Gowalla's dataset, on the other hand, includes 736,148 check-ins made by users in California and Nevada in the period of Feb. 2009 to Oct. 2010 [208]. Every check-in contains a user id, POI id, time, and geographical coordinates. For a fair comparison, we also use the preprocessed data as in [208]. Users who have less than 5 check-ins, as well as POIs with less than 5 check-ins are removed from the datasets. Finally, we have 194,108 check-ins on Foursquare's and 456,988 check-ins on Gowalla's. We used the geographical coordinates of POIs to retrieve their corresponding categories such as bar, pizza place. More details are listed in Table 4.1. Notice that multiple check-ins refer to the check-ins that were made by a particular user to a particular POI more than once. As shown in Table 4.1, 45.51% of check-ins on the Foursquare's dataset refer to users visiting POIs more than once. In Gowalla's, we observe fewer multiple check-ins, namely, 32.69% of all check-ins.

4.2.2 Time-Dependency of User Activities and Interests

Many studies analyzed users activity patterns over time [86, 200, 208, 209]. We conduct a long-term user activity analysis, namely we study the user behavior over several months. This monthly analysis allows us to realize the long-term shift of users interest. Users' interests could evolve over time while sustaining their personal patterns. For instance, a teenager who reaches the legal age of drinking often starts going to bars (long-term interest shift), whereas an adult user often chooses to travel on Easter (personal pattern). Figure 4.1 depicts the distribution of check-ins of both datasets every month. There is a significant shift

¹Available at <http://www.ntu.edu.sg/home/gaocong/data/poidata.zip>

Table 4.1. General statistics of the datasets

	Foursquare's	Gowalla's
# of users	2,231	10,162
# of POIs	5,596	24,250
# of check-ins	194,108	456,988
Avg. POIs per user	45.57	30.356
Avg. users per POI	18.90	12.69
Multiple check-ins	45.51%	32.69%
Density	0.0081	0.0012

of activity on each month. For instance, we observe that the number of check-ins increases from Feb. to Jul. in both datasets. Next, we report further analysis only on Gowalla's, as we observed similar attributes when analyzing the Foursquare's dataset. Figure 4.2 shows the popularity of the top 8 venue categories. As we can see, before Oct. 2009 we observe dramatic shift of popularity. However, after Oct. 2009, the top categories exhibit a rather more stable popularity pattern as the dataset grows. Interestingly, we still can observe popularity shifts between "Coffee Shop," "American," "Mexican," "City Park," and "Asian." This suggests that users also exhibit a time-dependent pattern in visiting popular categories of POIs.

Figure 4.3 shows the activity of two samples of users in a long period. It also shows how the popularity of two samples of POI categories changes over time. Note that, all the plots of Figure 4.3 illustrate the distribution of each user or category over time, that is, for each user or category, summing all the values of Y over time (X) equals to 1. We compute the variance of check-ins for both users and categories. Then, we pick the ones with the lowest in Figure 4.3a & c and the ones with the highest variances in Figure 4.3b & d. Our aim is to pick some representative user and categories from either end of the spectrum. As we can see, even users who are supposed to be the most stable (Figure 4.3a) exhibit very time-dependent check-in patterns. It is worth noting that we ignored users with less than 50 check-ins in plotting Figure 4.3b, as those users mostly appear once in the dataset and hence are listed among the top variant users. However, we still observe that the users with high check-in variance are those who appear a few times and have less regular check-in behavior. This suggests

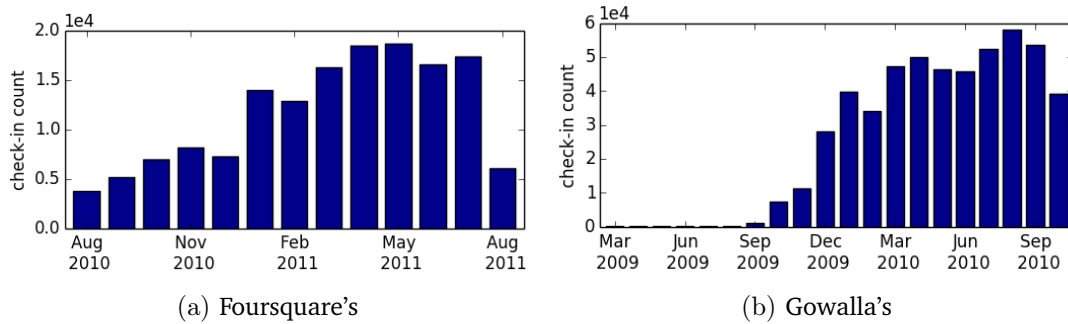


Figure 4.1. Number of check-ins per month on Foursquare's and Gowalla's.

that the more active a user is, the less variant her check-in pattern is. We also observed *a negative correlation between the users' check-in variance and quantity* (Spearman: -0.5583) which supports our assumption. As for the POI categories, Figure 4.3c depicts the least variant categories. First, all of the categories relate to users' daily activities, for example going to a coffee shop or sandwich store. These categories are more time-independent since they are less affected by seasonal changes or weather conditions. Similar to what we observed for users, more popular categories are also less time-variant. In fact, we found *a negative correlation between POI categories variance and popularity* (Spearman: -0.8411). On the other hand, the more time-variant categories (Figure 4.3d) are less popular and they depend mainly on weather and political events. For example, "Ski Shop" only appears during winter and "Democratic Event" appears as a political campaign takes place.

All our observations suggest that there is an underlying temporal pattern in check-in activities from the perspective of both users and place categories. We argued that check-in variance reveals meaningful information and thus we design our model focusing on these aspects.

4.2.3 Users' Multiple Check-ins

As we discussed earlier in Section 4.1, a good indicator of a user's preference is the fact that they visit the same POI multiple times. A multiple check-in between a user ρ and a venue l occurs if ρ visits l more than once. Other studies assumed multiple check-ins as an implicit feedback and demonstrated its effectiveness [125]. We also observed that a considerable amount of check-ins are multiple check-ins (Table 4.1). We randomly picked 100 users from the 500 most active users of Gowalla's. Figure 4.4 shows the quantity of single check-ins

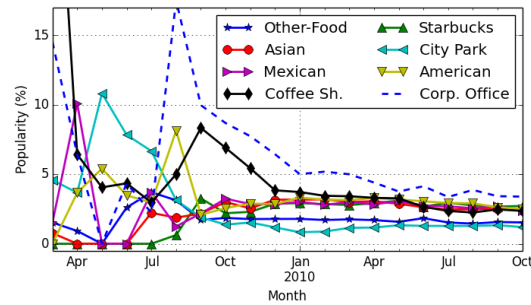


Figure 4.2. Popularity of the top-8 categories over time on Gowalla's (best viewed in color).

as well as multiple check-ins of the sample users. As we can see, users spend a considerable amount of their time visiting POIs they have visited before. In fact, multiple check-ins constitute 20% of check-ins for an average user on Gowalla's (with a standard deviation of 21). According to Figure 4.4, the more active a user is, the more multiple check-ins they have. We calculated the correlation between the number of check-ins and the percentage of multiple check-ins per user. We observe that there is a positive correlation (Spearman: +0.4257) between the two variables, supporting our assumption. Based on these observations, we confirm that multiple check-ins provide valuable information regarding users' preference and since these account for a considerable amount of the datasets, we design our model with a focus on them.

4.2.4 Remarks

In this section, we present the main findings of our analysis. As observed in Section 4.2.2, users exhibit a long-term shift in their check-in behavior. As a consequence, POIs witness a shift in their popularity. However, many of these shifts are natural because users grow old and their habits evolve. Also, even though some types of POIs exhibit less time-sensitivity, many POIs are highly dependent on temporal phenomena such as seasonal changes or political campaigns. In addition, over a longer period of time, less active users exhibit more variance in their check-in behavior. The same behavior is also observed for POIs, that is, less popular POIs suffer from more variance in terms of users' check-ins. Thus, it is crucial to take into consideration how variant a user or POI has been in the past while learning the model.

According to 4.2.3, there is a positive correlation between the activity of a user and the number multiple check-ins they have had. This also means that

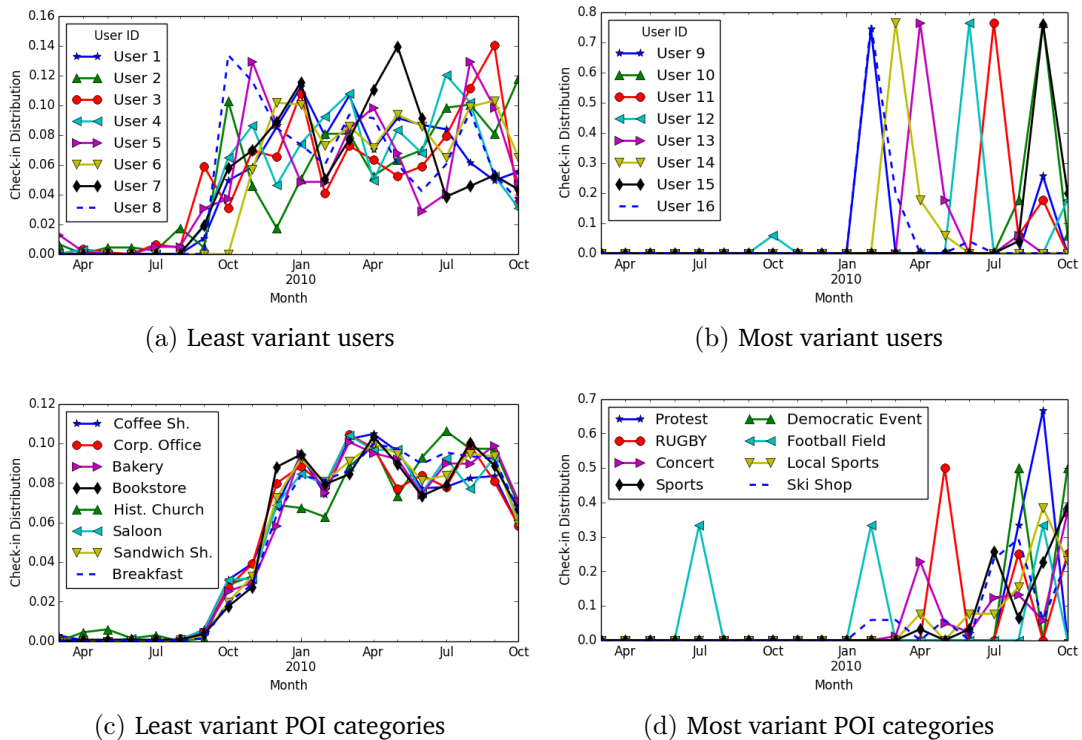


Figure 4.3. Check-in distribution of users and POI categories over time in Gowalla's. Figures a & c depict the least time-variant users and POI categories, respectively. Figures b & d, in contrast, show the distribution of the most time-variant users and POI categories, respectively (best viewed in color).

a user who has visited the same POI several times in the past is more likely to visit the same POI in the future. Moreover, single check-ins are already crucial and should be considered in the model to account for users' preferences as they choose to visit a particular POI when they could have chosen other venues in the same neighborhood. This gives us a rough estimate of their interest and preferences while multiple check-ins are more accurate indicators of users' interest. Thus, both single and multiple check-ins need to be involved while learning the model.

4.3 Proposed Method

Following the remarks based on our analysis in Section 4.2, in this section, we present here our model. We first introduce the notations of our model. Let

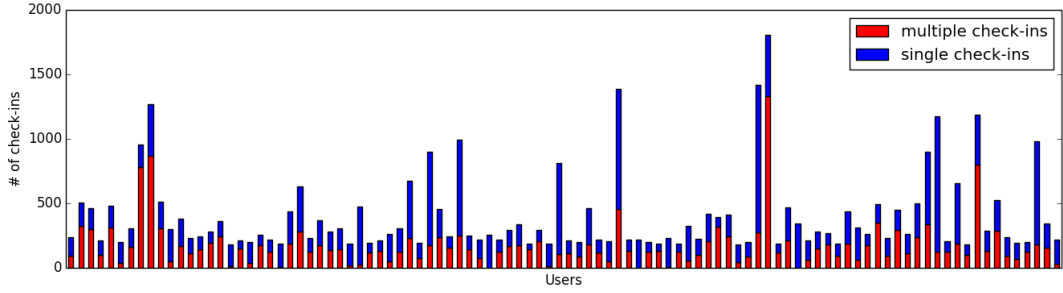


Figure 4.4. Check-in histogram of 100 randomly-sampled users from the 500 most active users of Gowalla’s. For each user, the red bar denotes the number of multiple check-ins, while the blue bar denotes the number of single check-ins.

$\mathcal{P} = \{\rho_1, \dots, \rho_n\}$ be the set of n users and $\mathcal{L} = \{l_1, \dots, l_m\}$ be the set of m POIs. As an implicit feedback, we consider POIs that each user ρ_i has visited, \mathcal{L}_i , as “relevant” and all other unvisited POIs as “irrelevant” in the neighborhood where the user has visited all relevant items. Also, we consider POIs with multiple check-ins as “more relevant.” Therefore, we define \mathcal{L}_i^+ as the set of relevant POIs, \mathcal{L}_i^- as the set of irrelevant POIs, and \mathcal{L}_i^* the set of “more-relevant” POIs for user ρ_i . We also define $n^+ = |\mathcal{L}_i^+|$, $n^- = |\mathcal{L}_i^-|$, and $n_i^* = |\mathcal{L}_i^*|$.

Our aim is to define a ranking function $f_i(l)$ for each user ρ_i to rank more-relevant POIs higher than relevant POIs, and relevant POIs higher than irrelevant POIs. Moreover, we aim to incorporate the influence of POIs that are located close to each other. Let $U \in \mathbb{R}^{d \times n}$ be the latent factor for users and $V \in \mathbb{R}^{d \times m}$ be the latent factor for POIs. It is worth noting that \mathbf{u}_i corresponds to ρ_i and \mathbf{v}_j corresponds to l_j . The ranking function for the i^{th} user ρ_i and the j^{th} POI l_j would be $f_i(l_j) = \mathbf{u}_i^T \mathbf{v}_j$.

We design a two-phase objective function, where the first phase constructs ranking functions $f_i(l)$ such that relevant POIs are ranked higher than irrelevant POIs. In the second phase, functions $f_i(l)$ are updated to rank more-relevant POIs higher than relevant POIs. In Section 4.3.1, we first describe how we compute the distance between two POIs and in Section 4.3.2 we explain the first phase of the objective function in which the geographical influence is also incorporated. In Section 4.3.3, we give details on the second phase of the objective function. In Section 4.3.4, we describe the proposed time-sensitive regularizer which takes into consideration the users’ long-term behavioral patterns, and in Section 4.3.5 we present an overview of the proposed algorithm and how both phases of the objective function are optimized jointly.

4.3.1 Geographical Similarity

We compute the geographical similarity between two POI to incorporate the geographical context while characterizing the user’s geographical preferences. The similarity is inversely proportional to the distance between two POIs. Inspired by relevant studies [5, 17, 125, 126] where a simple geographical measure improved the models significantly, we use the Haversine formula to compute the angular distance between l_i and l_j :

$$h_{ij} = 2 \arcsin \left(\sqrt{\sin^2(\Delta\phi_{ij}/2) + \cos\phi_i \times \cos\phi_j \times \sin^2(\Delta\eta_{ij}/2)} \right),$$

where ϕ_i and ϕ_j are latitudes of l_i and l_j in radian, respectively. Accordingly, η_i and η_j are longitudes of l_i and l_j in radian. Then we calculate the geographical similarity between l_i and l_j as follows:

$$g_{ij} = \frac{1}{1 + (h_{ij} \times R)},$$

where R is the earth’s radius ($R = 6,371\text{KM}$). In the following section, we demonstrate how d_{ij} is incorporated in the our proposed method.

4.3.2 Phase 1: Visited vs. Unvisited POIs

In the first phase, we focus on ranking higher POIs that a user has visited (no matter how many times) than the ones they did not visit. Formally, we aim at ranking \mathcal{L}_i^+ higher than \mathcal{L}_i^- . More specifically, our goal is to rank the POIs with emphasis on the top of the list. Moreover, we take into consideration the geographical distance between POIs. Building ranking functions that incorporate the distance between POIs also allows us to model latent associations between users living in the same neighborhood, who would not be associated with each other in a traditional CR setting. This happens because our method takes into account venues’ distances as it updates the user and item latent matrices.

Let $H_i(l_j^-)$ be the “height” of an irrelevant venue:

$$H_i(l_j^-) = \sum_{k \in \mathcal{L}_i^+} \left(\frac{\mathbf{1}_{[f_i(l_k^+) \leq f_i(l_j^-)]}}{1 + \alpha \exp(g_{kj})} \right), \quad (4.1)$$

where α is the weight of geographical influence and $\mathbf{1}_{[\cdot]}$ is an indicator function. Note that α controls the model’s bias towards POIs in the same neighborhood and can be used to prevent the “Harry Potter” problem [115]. Dividing the indicator

function by G allows the model to incorporate the geographical distances into the model while constructing the height for irrelevant items. For example, if an irrelevant item is ranked higher than a relevant item, but they are very close then the denominator will be higher, reducing the height of the irrelevant POI. The objective function should aim at minimizing H_i for all given POIs. A lower value of H_i also means that there are fewer irrelevant POIs ranked higher than relevant ones. From an optimization perspective, indicator functions are not convex. Therefore, we use the logistic loss of the difference between the two functions as a convex upper bound surrogate. For reading simplicity we define

$$G_\alpha(i, j) = 1 + \alpha \exp(g_{ij}) .$$

We define the difference between the k^{th} POI and the j^{th} as follows:

$$\delta_i(k, j) = \mathbf{u}_i^T \left((\mathbf{v}_k - \mathbf{v}_j) / G_\alpha(k, j) \right) . \quad (4.2)$$

Therefore, the surrogate height function $H'_i(\ell_j^-)$ becomes:

$$H'_i(\ell_j^-) = \sum_{k \in \mathcal{L}_i^+} \log \left[1 + \exp \left(-\delta_i(k, j) \right) \right] , \quad (4.3)$$

where $\ell(\delta) = \log(1 + \exp(-\delta))$ is the logistic loss of δ . We consider ℓ_2 -norm of H'_i as the objective function, following [167]. Therefore, the objective function is:

$$\begin{aligned} R(U, V) &= \sum_{i=1}^m \frac{1}{n_i} \sum_{j \in \mathcal{L}_i^-} (H'_i(\ell_j^-))^2 = \sum_{i=1}^m \frac{1}{n_i} \times \\ &\sum_{j \in \mathcal{L}_i^-} \left(\sum_{k \in \mathcal{L}_i^+} \log \left(1 + \exp \left(-\mathbf{u}_i^T \left[(\mathbf{v}_k - \mathbf{v}_j) / G_\alpha(k, j) \right] \right) \right) \right)^2 . \end{aligned} \quad (4.4)$$

For solving the above optimization problem, we use a gradient-descent-based alternating optimization algorithm. We first keep V fixed and update U , and then keep U and update V . Therefore, the update rules of the $t+1$ iteration are:

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \gamma \nabla_{\mathbf{u}_i} R(U^t, V^t), \forall i = 1, \dots, n , \quad (4.5)$$

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \gamma \nabla_{\mathbf{v}_j} R(U^{t+1}, V^t), \forall j = 1, \dots, m . \quad (4.6)$$

For reading simplicity we define

$$\theta(k, j) = \left(1 + \exp(\delta(k, j)) \right) ,$$

and the gradients of $R(U, V)$ with respect to \mathbf{u}_i and \mathbf{v}_j are computed as follows:

$$\begin{aligned} & \nabla_{\mathbf{u}_i} R(U, V) \\ &= \frac{2}{n_i} \sum_{j \in \mathcal{L}_i^-} \left(H'_i(l_j^-) \sum_{k \in \mathcal{L}_i^+} (\mathbf{v}_j - \mathbf{v}_k) / (G_\alpha(k, j) \theta(k, j)) \right), \\ & \nabla_{\mathbf{v}_j} R(U, V) \\ &= \sum_{i \in \mathcal{P}_j^-} \frac{2}{n_i} \sum_{h \in \mathcal{L}_i^-} \left(H'_i(l_h^-) \sum_{k \in \mathcal{L}_i^+} \mathbf{u}_i / (G_\alpha(k, h) \theta(k, h)) \right) \\ & \quad - \sum_{i \in \mathcal{P}_j^+} \frac{2}{n_i} \sum_{h \in \mathcal{L}_i^-} \left(H'_i(l_h^-) \sum_{k \in \mathcal{L}_i^+} \mathbf{u}_i / (G_\alpha(k, h) \theta(k, h)) \right). \end{aligned}$$

with \mathcal{P}_j^+ being the set of users who have visited l_j and \mathcal{P}_j^- the set of users who have not visited l_j .

4.3.3 Phase 2: Multiple vs. Single Check-ins

In the second phase, we focus on ranking POIs with multiple check-ins (i.e., more-relevant POIs) higher than the ones with single check-in (i.e., relevant POIs). As we observed in Section 4.2.3, there is a positive correlation between the number of multiple check-ins and total number check-ins. Therefore, it is crucial to take into consideration the fact that the POIs that users have visited more often in the past are more relevant. Formally, let $\mathcal{L}_i^{1+} = \mathcal{L}_i^+ - \mathcal{L}_i^*$ be the set of POIs that the i^{th} user has visited exactly once. Our goal is to rank \mathcal{L}_i^* higher than \mathcal{L}_i^{1+} . Let $\Pi_i(l_j^*)$ be the “reverse height” of a more-relevant venue, that is:

$$\Pi_i(l_j^*) = \sum_{k \in \mathcal{L}_i^{1+}} \mathbf{1}_{[f_i(l_k^*) \leq f_i(l_j^{1+})]}. \quad (4.7)$$

Lower values of Π_i mean that there are fewer relevant POIs ranked higher than the more-relevant ones. Similarly to the first phase in Section 4.3.2, we use a logistic loss as the surrogate:

$$\Pi'_i(l_j^{1+}) = \sum_{k \in \mathcal{L}_i^*} \log(1 + \exp(-\mathbf{u}_i^T (\mathbf{v}_k - \mathbf{v}_j))). \quad (4.8)$$

However, Π'_i is not easy to be optimized using typical ranking loss, like DCG. Hence, we reformulate the objective functions as follows:

$$\sum_{k \in \mathcal{L}_i^*} \log(1 + \Pi_i(l_k^*)).$$

Then, the objective function $R_{\Pi}(U, V)$ of reverse height becomes:

$$R_{\Pi}(U, V) = \sum_{i=1}^m \frac{1}{n_i} \sum_{j \in \mathcal{L}_j^*} \log(1 + \Pi'_i(l_k^*)) = \sum_{i=1}^m \frac{1}{n_i} \times \sum_{j \in \mathcal{L}_j^*} \log \left(1 + \sum_{k \in \mathcal{L}_i^{1+}} \log \left(1 + \exp(-\mathbf{u}_i^T(\mathbf{v}_k - \mathbf{v}_j)) \right) \right). \quad (4.9)$$

We optimize $R_{\Pi}(U, V)$ similarly to Section 4.3.2, that is, we first keep V fixed and update U , then keep U updating V . Therefore, we consider the following update rules:

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \gamma \nabla_{\mathbf{u}_i} R_{\Pi}(U^t, V^t), \forall i = 1, \dots, n, \quad (4.10)$$

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \gamma \nabla_{\mathbf{v}_j} R_{\Pi}(U^{t+1}, V^t), \forall j = 1, \dots, m. \quad (4.11)$$

Similarly, the gradients are defined as follows:

$$\begin{aligned} & \nabla_{\mathbf{u}_i} R_{\Pi}(U, V) \\ &= \frac{1}{n_i} \sum_{j \in \mathcal{L}_j^*} \left(\frac{1}{1 + \Pi'_i(l_j^*)} \sum_{k \in \mathcal{L}_i^{1+}} ((\mathbf{v}_j - \mathbf{v}_k) / (1 + \exp(\delta_i(k, j)))) \right), \end{aligned}$$

$$\begin{aligned} & \nabla_{\mathbf{v}_j} R_{\Pi}(U, V) \\ &= \sum_{i \in \mathcal{P}_j^{1+}} \frac{1}{n_i} \sum_{h \in \mathcal{L}_i^{1+}} \left(\frac{1}{1 + \Pi'_i(l_h^*)} \sum_{k \in \mathcal{L}_i^*} (\mathbf{u}_i / (1 + \exp(\delta_i(k, h)))) \right) \\ & - \sum_{i \in \mathcal{P}_j^*} \frac{1}{n_i} \sum_{h \in \mathcal{L}_i^{1+}} \left(\frac{1}{1 + \Pi'_i(l_h^*)} \sum_{k \in \mathcal{L}_i^*} (\mathbf{u}_i / (1 + \exp(\delta_i(k, h)))) \right), \end{aligned}$$

where we define \mathcal{P}_j^{1+} the set of users who have visited l_j only once and \mathcal{P}_j^* the set of users who have visited l_j multiple times.

4.3.4 Time-Sensitive Regularizer

As we discussed in Section 4.2.2, the long-term temporal activity patterns of both users and POIs should be taken into account. One way to account for the activity patterns of users and the popularity of POIs is to consider how variant they are over time. For example, if a POI is a coffee shop and receives approximately the same number of people every month, it is more likely that it receives the same number of users in the next month. Whereas, POIs like ski shop are only popular

during the ski season. In particular, we observed in Figure 4.3d that the popularity of certain POI categories are highly time-dependent. Here, we propose to incorporate a novel time-sensitive regularizer into the objective function of both of our objective phases in (4.4) and (4.9). Adding a regularizer that is calculated for each user and POI based on their past activities enables us to model the time-sensitivity of users and POIs. The main goal here is to penalize those users and POIs which are less stable. A more stable user or POI is one that exhibits less activity variance over time. This regularizer is defined based on our extensive analysis and observation in Section 4.2.2 where we observed that POIs that are less popular are more time-sensitive. We also had a similar observation for users, where we observed that less active users exhibit less stability in their check-in behavior.

Let $\sigma^{2,U} \in \mathbb{R}^{1 \times n}$ be the variance vector for users, where $\sigma_i^{2,U}$ is the activity variance of the i^{th} user ρ_i . For each user ρ_i , we count the number of check-ins per month and normalize the values. Then, calculating the variance of the monthly check-ins of ρ_i produces $\sigma_i^{2,U}$. Similarly, let $\sigma^{2,V} \in \mathbb{R}^{1 \times m}$ be the variance vector for POIs, with $\sigma_j^{2,V}$ being the popularity variance of the j^{th} POI l_j . Note that we calculate the variance of POIs based on their corresponding categories since we observed more meaningful popularity patterns with respect to the categories. The time-sensitive regularizer parameter for users and POIs are calculated as follows:

$$\Lambda^U = \lambda \log(1 + \exp(-\sigma^{2,U})), \quad (4.12)$$

$$\Lambda^V = \lambda \log(1 + \exp(-\sigma^{2,V})). \quad (4.13)$$

It is worth noting that we consider the logistic function of variances to prevent underflow and take the hyper parameter λ as a controlling parameter to prevent the model from overfitting. Ultimately, we add the time-sensitive regularizer to the objective functions of our two phases in (4.4) and (4.9). Thus, when updating \mathbf{u}_i , we add the regularizer term $\Lambda_i^U \mathbf{u}_i$, and for updating \mathbf{v}_j we add the regularizer term $\Lambda_j^V \mathbf{v}_j$.

4.3.5 Joint Two-Phase Collaborative Ranking Algorithm

Algorithm 1 presents the proposed joint two-phase collaborative ranking method. Line 2 initializes the factor matrices randomly. θ is initialized at line 3, summing up the values of the two phases of our objective function, namely, $R(U^{t+1}, V^{t+1})$ and $R_{\Pi}(U^{t+1}, V^{t+1})$. The joint optimization of the two phases is done between lines 4 and 12. As we see, in every iteration, \mathbf{u}_i and \mathbf{v}_j are first updated according to (4.5) and (4.6) (lines 7 and 8) to push visited POIs higher in the ranking.

Algorithm 1: The Joint Two-Phase Collaborative Ranking Algorithm (JTCR).

Input: $\mathcal{P}, \mathcal{L}, \maxIter, \{\lambda, \gamma, \alpha, \epsilon, d\}$
Output: $U_{\text{out}}, V_{\text{out}}$

- 1 $t \leftarrow 0$
- 2 Initialize U^{t+1}, V^{t+1}
- 3 $\theta^{t+1} \leftarrow R(U^{t+1}, V^{t+1}) + R_{\Pi}(U^{t+1}, V^{t+1}), \theta^t = \frac{\theta^{t+1}}{2}$
- 4 **while** ($\text{abs}(\theta^{t+1} - \theta^t) > \epsilon$) \wedge ($t < \maxIter$) **do**
- 5 $t \leftarrow t + 1$
- 6 // Phase 1
- 7 Update $\mathbf{u}_i^{t+1}, \forall i = 1, \dots, n$ Eq. (4.5)
- 8 Update $\mathbf{v}_j^{t+1}, \forall j = 1, \dots, m$ Eq. (4.6)
- 9 // Phase 2
- 10 Update $\mathbf{u}_i^{t+1}, \forall i = 1, \dots, n$ Eq. (4.10)
- 11 Update $\mathbf{v}_j^{t+1}, \forall j = 1, \dots, m$ Eq. (4.11)
- 12 $\theta^{t+1} \leftarrow R(U^{t+1}, V^{t+1}) + R_{\Pi}(U^{t+1}, V^{t+1})$
- end**
- 13 $U_{\text{out}} \leftarrow U^{t+1}, V_{\text{out}} \leftarrow V^{t+1}$

Each iteration is then followed by optimizing \mathbf{u}_i and \mathbf{v}_j according to (4.10) and (4.11) (lines 10 and 11), respectively. Therefore, U and V factor matrices are optimized jointly to push visited POIs higher than unvisited POIs and multiple visited POIs higher than single visited POIs simultaneously. After convergence, the final values of the latent factor matrices are stored at line 13. Note that the proposed time-sensitive regularizer is applied at lines 7, 8, 10, and 11. Also, the geographical influence is applied at lines 7 and 8. One can argue that employing a two-phase learning strategy might be computationally expensive. However, since in the second phase we only focus on the POIs that each user has checked in, the optimization algorithm does not add a substantial overhead to the whole system. In fact, for each user the complexity of one iteration is $\mathcal{O}(n_i^{1+} n_i^*)$, which is very small compared to the first phase, which is $\mathcal{O}(n_i^- n_i^+)$.

4.4 Experimental Setup

In this section, we evaluate the performance of our model compared with state-of-the-art methods and study the impact of different parameters on the performance of our model.

4.4.1 Data

We evaluate our method on two real-world datasets, namely, Foursquare’s and Gowalla’s. Both datasets were provided by the authors of [208]. The statistical details of the datasets are listed in Table 4.1. We take the first 70% of the data for each user as the training set, 10% as the validation set, and the remaining 20% as the test set, following the evaluation protocol of [219].

4.4.2 Metrics

We compare the performance of our model in terms of *Precision at k* (P@k) and *Normalized Discounted Cumulative Gain at k* (nDCG@k). Let $L_{ch}(\rho)$ be the set POIs that a user has visited in the test set and $L_{rec}^k(\rho)$ be the set of recommended POIs of size k . P@k(ρ) for a user ρ is defined as $P@k(\rho) = (|L_{ch}(\rho) \cap L_{rec}^k(\rho)|)/(k)$ and P@k for the whole dataset is the average P@k(ρ) for all the users in the test set.

To calculate nDCG@k, we need to define relevance values in the test set. Following the same strategy of Section 4.3, we define a three-level relevance for each POI based on the frequency of check-in for a particular user:

$$rel(l, \rho) = \begin{cases} 2 & \text{if } \rho \text{ visited } l \text{ multiple times} \\ 1 & \text{if } \rho \text{ visited } l \text{ only once} \\ 0 & \text{if } \rho \text{ did not visit } l . \end{cases}$$

Therefore, nDCG@k(ρ) for a given user ρ is defined as follows:

$$DCG@k(\rho) = \sum_{r=1}^k \frac{2^{rel(l_r, \rho)} - 1}{\log_2(r + 1)},$$

$$nDCG@k(\rho) = \frac{DCG@k(\rho)}{IDCG@k(\rho)},$$

where l_r is the POI at the r^{th} rank and IDCG@k(ρ) is the ideal DCG@k value for user ρ , that is, the highest possible value for DCG@k. The reported values of nDCG@k are the average of the nDCG@k(ρ) values for all the users in the test set. We report the values of nDCG@k and P@k for three values of k, namely 5, 10, 20.

4.4.3 Compared Methods

We compare our **Joint Two-Phase Collaborative Ranking (JTTCR)** model with approaches that consider geographical influence for POI recommendation and approaches based on collaborative ranking with emphasis on ranking relevant items higher. Also, we include two variations of the proposed JTTCR to demonstrate the effectiveness of different elements of our algorithm. Note that for each model, we find the optimum set of parameters using the validation set and report the mean and standard deviation of results of 5 different runs with the same parameters. We compare our JTTCR model with the following methods:

- **JTTCR-Phase1** reports the performance of the first phase of JTTCR. We include this model as a baseline to demonstrate the effectiveness of the first phase of JTTCR and the significance of the second phase of the algorithm.
- **JTTCR-NoVar** reports the result of our proposed JTTCR without using the time-sensitive regularizer. Instead, we use $\lambda/2(|U|^2 + |V|^2)$ as the regularizer. Our goal is to demonstrate the effectiveness of the time-based regularizer.
- **JTTCR-NoGeo** reports the result of our proposed JTTCR without applying the geographical influence (i.e., $\alpha = 0$).
- **WRMF** [101] proposes an MF method for item prediction from implicit feedback. It is an adaptation of SVD, minimizing the square-loss.
- **IRenMF** [131] is based on weighted MF [143] exploiting two levels of geographical neighborhood characteristics: nearest neighboring locations share more similar user preferences, while locations in the same geographical region may share similar user preferences.
- **GeoMF** [126] augments users' and venues' latent factors in the factorization model with activity area vectors of users and influence area vectors of venues, respectively.
- **Rank-GeoFM** [125] is a ranking-based MF model that includes the geographical influence of neighboring venues while learning users' preference rankings for venues.
- **Rank-GeoFM-NoGeo** reports the result of Rank-GeoFM without considering the geographical influence.

- **RH-Push / Inf-Push / P-Push** [60] are three push CR models based on reverse height, infinite, and p -norm. For each user, we considered the venues they visited as positive training samples and selected k venues randomly as negative training samples.

We aim to compare the performance of JTTCR against state-of-the-art methods in POI recommendation that consider recommendation as a ranking problem and the ones that do not. Also, it is crucial to compare our method with approaches that incorporate geographical influence into the model. The other set of methods is based on CR. Our aim is to demonstrate the effectiveness of our two-phase regularized CR in comparison with other CR baselines.

4.5 Results and Discussion

In this section, we evaluate the performance of our proposed method with baseline methods.

4.5.1 Performance Comparison

Tables 4.2, 4.3, 4.4, and 4.5 report the performance of our method compared with 11 baselines in terms of nDCG@k and P@k for Foursquare’s and Gowalla’s. Based on the results we observe that our proposed JTTCR significantly outperforms all the baseline methods on both datasets with respect to both nDCG@k and P@k. It is worth noting that the improvement is achieved for all values of k .

Moreover, Rank-GeoFM performs best among the geographical-based methods, as Rank-GeoFM propagates geographical influences using the constructed graph, which confirms that geographical neighborhood is a major factor for recommendation. Rank-GeoFM considers the implicit feedback while training the model similar to us, however, as we observe our two-phase collaborative ranking approach beats Rank-GeoFM indicating the effectiveness of our approach. Moreover, JTTCR outperforms Rank-GeoFM by a large margin on both datasets. It is worth noting that JTTCR beats all geographical-based methods in terms of both nDCG@k and P@k for all different values of k in both datasets. GeoMF and IReNMF do not consider POI recommendation as a ranking problem. Hence, they attempt to optimize the overall error rate which proves to be less effective for POI recommendation mainly because the users are only interested in top k recommended POIs. Consequently, JTTCR beats GeoMF and IReNMF with a large margin.

Table 4.2. Performance evaluation on Foursquare’s in terms of nDCG@k.

	nDCG@5	Δ	nDCG@10	Δ	nDCG@20	Δ
JTCR	0.0639	-	0.0529	-	0.0394	-
JTCR-Phase1	0.0534 [†]	-16.43%	0.0436 [†]	-17.58%	0.0339 [†]	-13.96%
JTCR-NoVar	0.0605 [†]	-5.32%	0.0497 [†]	-6.05%	0.0375 [†]	-4.82%
JTCR-NoGeo	0.0613 [†]	-4.07%	0.0494 [†]	-6.62%	0.0381	-3.3%
WRMF	0.0248 [†]	-61.19%	0.0210 [†]	-60.3%	0.0178 [†]	-54.82%
GeoMF	0.0422 [†]	-33.96%	0.0336 [†]	-36.48%	0.0250 [†]	-36.55%
IRenMF	0.0430 [†]	-32.71%	0.0348 [†]	-34.22%	0.0286 [†]	-27.41%
Rank-GeoFM	0.0438 [†]	-31.46%	0.0359 [†]	-32.14%	0.0277 [†]	-29.70%
Rank-GeoFM-NoGeo	0.0418 [†]	-34.59%	0.0314 [†]	-40.64%	0.0234 [†]	-40.61%
RH-Push	0.0251 [†]	-60.72%	0.0187 [†]	-64.65%	0.0137 [†]	-65.23%
Inf-Push	0.0433 [†]	-32.24%	0.0361 [†]	-31.76%	0.0302 [†]	-23.35%
P-Push	0.0423 [†]	-33.8%	0.0309 [†]	-41.59%	0.0218 [†]	-44.67%

Statistically significant differences with JTCR are denoted by [†] for $p < 0.05$ in paired t-test. Δ values express the relative difference, compared to JTCR. For each model we report the mean and standard deviation of 5 different runs.

Table 4.3. Performance evaluation on Foursquare’s in terms of P@k.

	P@5		P@10		P@20	
	P	Δ	P	Δ	P	Δ
JTCR	0.0591	-	0.0456	-	0.0303	-
JTCR-Phase1	0.0462 [†]	-21.83%	0.0357 [†]	-21.71%	0.0260 [†]	-14.19%
JTCR-NoVar	0.0536 [†]	-9.31%	0.0414 [†]	-9.21%	0.0282 [†]	-6.93%
JTCR-NoGeo	0.0551 [†]	-6.77%	0.0411 [†]	-9.87%	0.0292	-3.63%
WRMF	0.0224 [†]	-62.1%	0.0181 [†]	-60.31%	0.0151 [†]	-50.17%
GeoMF	0.0385 [†]	-34.86%	0.0281 [†]	-38.38%	0.0186 [†]	-38.61%
IRenMF	0.0385 [†]	-34.86%	0.0280 [†]	-38.60%	0.0219 [†]	-27.72%
Rank-GeoFM	0.0397 [†]	-32.83%	0.0304 [†]	-33.33%	0.0215 [†]	-29.04%
Rank-GeoFM-NoGeo	0.0339 [†]	-42.64%	0.0229 [†]	-49.78%	0.0157 [†]	-48.18%
RH-Push	0.0214 [†]	-63.79%	0.0140 [†]	-69.3%	0.0094 [†]	-68.98%
Inf-Push	0.0397 [†]	-32.83%	0.0313 [†]	-31.36%	0.0217 [†]	-28.38%
P-Push	0.0326 [†]	-44.84%	0.0211 [†]	-53.73%	0.0132 [†]	-56.44%

Statistically significant differences with JTCR are denoted by [†] for $p < 0.05$ in paired t-test. Δ values express the relative difference, compared to JTCR. For each model we report the mean and standard deviation of 5 different runs.

Table 4.4. Performance evaluation on Gowalla’s in terms of nDCG@k.

	nDCG@5	Δ	nDCG@10	Δ	nDCG@20	Δ
JTCR	0.1158	-	0.0854	-	0.0633	-
JTCR-Phase1	0.1090 [†]	-4.32%	0.0823	-3.04%	0.0607	-2.53%
JTCR-NoVar	0.1092 [†]	-5.79%	0.0802 [†]	-5.85%	0.0593 [†]	-7.42%
JTCR-NoGeo	0.1099 [†]	-5.09%	0.0823	-3.63%	0.0608	-3.95 [†] %
WRMF	0.0620 [†]	-46.46%	0.0523 [†]	-38.76%	0.0425 [†]	-32.86%
GeoMF	0.0604 [†]	-47.84%	0.0495 [†]	-42.04%	0.0374 [†]	-40.92%
IRenMF	0.0606 [†]	-47.67%	0.0499 [†]	-41.57%	0.0389 [†]	-38.55%
Rank-GeoFM	0.0593 [†]	-48.79%	0.0525 [†]	-38.52%	0.0451 [†]	-28.75%
Rank-GeoFM-NoGeo	0.0675 [†]	-41.71%	0.0514 [†]	-39.81%	0.0387 [†]	-38.86%
RH-Push	0.0985 [†]	-14.94%	0.0765 [†]	-10.42%	0.0569 [†]	-10.11%
Inf-Push	0.1090 [†]	-5.87%	0.0803 [†]	-5.97%	0.0585 [†]	-7.58%
P-Push	0.1026 [†]	-11.40%	0.0805 [†]	-5.74%	0.0596 [†]	-5.85%

Statistically significant differences with JTCR are denoted by [†] for $p < 0.05$ in paired t-test. Δ values express the relative difference, compared to JTCR. For each model we report the mean and standard deviation of 5 different runs.

Table 4.5. Performance evaluation on Gowalla’s in terms of P@k.

	P@5		P@10		P@20	
	P	Δ	P	Δ	P	Δ
JTCR	0.0949	-	0.0621	-	0.0425	-
JTCR-Phase1	0.0889 [†]	-6.32%	0.0596 [†]	-4.03%	0.0414	-2.59%
JTCR-NoVar	0.0865 [†]	-8.85%	0.0570 [†]	-8.21%	0.0378 [†]	-11.06%
JTCR-NoGeo	0.0866 [†]	-8.75%	0.0590 [†]	-4.99%	0.0401 [†]	-5.65%
WRMF	0.0556 [†]	-41.41%	0.0448 [†]	-27.86%	0.0346 [†]	-18.59%
GeoMF	0.0540 [†]	-43.10%	0.0415 [†]	-33.17%	0.0284 [†]	-33.18%
IRenMF	0.0545 [†]	-42.57%	0.0423 [†]	-31.88%	0.0305 [†]	-28.24%
Rank-GeoFM	0.0564 [†]	-40.57%	0.0472 [†]	-23.99%	0.0384 [†]	-9.65%
Rank-GeoFM-NoGeo	0.0549 [†]	-42.15%	0.0383 [†]	-38.33%	0.0269 [†]	-36.71%
RH-Push	0.0820 [†]	-13.59%	0.0588 [†]	-5.31%	0.0400 [†]	-5.88%
Inf-Push	0.0864 [†]	-8.96%	0.0569 [†]	-8.37%	0.0377 [†]	-11.29%
P-Push	0.0844 [†]	-11.06%	0.0573 [†]	-7.73%	0.0396 [†]	-6.82%

Statistically significant differences with JTCR are denoted by [†] for $p < 0.05$ in paired t-test. Δ values express the relative difference, compared to JTCR. For each model we report the mean and standard deviation of 5 different runs.

In addition, we observe that in most cases CR-based baseline methods, namely P-Push, Inf-Push, and RH-Push perform better than other baseline methods. This suggests that a CR-based approach leads to better performance for POI recommendation in general. However, we observe that JTCR outperforms all CR-based baselines. This indicates that all CR-based methods suffer from the sparsity problem while our two-phase CR strategy alleviates this problem by considering both visited and unvisited POIs in the same neighborhood. Also, none of the CR-based methods consider time in their ranking loss function. While it is important to consider POI recommendation as a ranking problem, it is also important to consider time to generate accurate recommendations (Section 4.2). Our proposed model beats the CR-based methods by a significant margin showing out time-sensitive regularizer (Section 4.3.4) based on the temporal behavior of users and the temporal popularity of POIs leads to a more accurate performance. Although the CR-based baseline approaches focus on the top of the ranked list, they fail to rank more-relevant POIs higher in the ranking. In fact, these methods consider a binary relevance between users and POIs. Higher nDCG@k values indicate that our model ranks POIs with multiple check-ins more accurately, compared with the CR-based baseline approaches. This suggests that our two-phase model ranks the POIs with higher relevance more effectively than the CR approaches by considering multi-level implicit user feedback. It is worth noting that the variants of our model also outperform most of the baselines. In particular, JTCR-NoVar outperforms all the baselines and JTCR-Phase1 performs better than most of the baselines, including P-Push. This is important, since it indicates that incorporating the temporal information together with geographical similarities improves the performance of this model when only the first phase is considered. Also, we see that JTCR-NoGeo beats all the baselines. Specifically, it performs better than Rank-GeoFM-NoGeo and other baselines that do not consider geographical information. This indicates the effectiveness of the proposed model even when the geographical influence is not considered.

4.5.2 Impact of the 2nd Phase

To study the effect of the second phase of JTCR, we compare the performance of JTCR when only the first phase is used (JTCR-Phase1) with the performance of JTCR when both phases are considered. As we can see in Tables 4.2 - 4.5, JTCR exhibits a significant improvement over JTCR-Phase1 in terms of all evaluation metrics for both datasets. This indicates that while JTCR-Phase1 is able to beat all other baselines, the second phase of the algorithm enables JTCR to model multiple check-ins more accurately. This validates the remark based on our analysis in

Section 4.2.4 which states that a user who has visited a POI multiple times in the past is likely to visit the same venue in the future. Moreover, while this remark based on our analysis applies to a user, it is also valid with respect to similar users. Therefore, in the second phase, similar users and their corresponding collaborative associations are mainly determined based on how similar they are in terms of multiple check-ins. This helps the model rank “more relevant” items higher and hence improves the accuracy of the model. Moreover, we observe a higher relative difference on Foursquare’s. According to Table 4.1, Foursquare’s consists of more multiple check-ins than Gowalla’s (45.51% as opposed to 32.69%) which suggests that the second phase can model multiple check-ins more effectively on the Foursquare’s dataset.

4.5.3 Impact of the Time-Sensitive Regularizer

Next, we discuss the effect of the time-sensitive regularizer. To this end, we compare the performance of JTCR without using the time-sensitive regularizer (JTCR-NoVar). As seen in Tables 4.2 & 4.3, a statistically significant improvement of JTCR over JTCR-NoVar is observed in terms of all evaluation metrics for Foursquare’s, suggesting that using the time-sensitive regularizer enables JTCR to place more relevant venues higher in the ranking. As for Gowalla’s, we also see significant improvements in Tables 4.4 & 4.5 indicating that our proposed time-sensitive regularizer improves the performance of JTCR by penalizing those users and POIs that exhibit less stability in their check-in and popularity, respectively. This validates the remark based on our analysis reported in Section 4.2.4 where we showed that there is a negative correlation between a POI’s popularity and its popularity variance. A similar observation was made for users. Based on this remark, we defined the time-sensitive regularizer to penalize those users and POIs that have been variant in the past. In other words, variant users are less probable to visit variant POIs and the introduced regularizer enables the model to take this into account while training.

4.5.4 Impact of the Geographical Influence

Here, we discuss the effect of the geographical influence. In order to do this, we compare the performance of JTCR without applying the geographical influence (JTCR-NoGeo). As seen in Tables 4.2 & 4.3, we observe a statistically significant improvement of JTCR over JTCR-NoGeo in terms of all evaluation metrics for Foursquare’s except for nDCG@20 and P@20. However, the significant improvement in terms of other evaluation metrics for Foursquare’s suggests that applying

the geographical influence enables JTCR to model users' geographical behavior and activities more effectively. As for Gowalla's, we see significant improvements in Tables 4.4 & 4.5 for all evaluation metrics expect nDCG@10 and nDCG@20. This indicates that the geographical influence improves the performance of JTCR by considering how users like POIs that are in the same neighborhood.

4.5.5 Impact of the Model Parameters

Next, we demonstrate the effect of the model's parameters. The results reported in the previous sections are achieved after the best parameter set was found on the validation set. We fixed the learning rate ($\gamma = 1 \times 10^{-4}$) for both datasets to ensure the generalization of our model.

In Figure 4.5 we study the effect of latent factors d on the performance of our model and report nDCG@5 while keeping other parameters fixed. As we can see in Figure 4.5a, the optimal number of latent factors d for Foursquare's is 80. For higher values of latent factors, nDCG@5 drops. Also, nDCG@5 drops when selecting lower values for d . We observe a similar behavior on Gowalla's in Figure 4.5b with the difference that the optimal number of latent factors is 90. For all other values of d , we observe a drop in the performance.

Furthermore, in order to study the effect of the regularizing control parameter (λ), we varied λ while keeping d and α fixed. As shown in Figure 4.6a, the best λ for Foursquare's is 1×10^{-4} , while according to Figure 4.6b, for Gowalla's, it is 1×10^{-4} . The performance of our model drops using different values of λ for both datasets. While lower performance achieved with lower values of λ indicate that the introduced regularizer is essential to avoid overfitting, higher values of λ also hurt the performance.

Next, in Figure 4.7 we study the effect of geographical influence weight α on the performance of our model and report nDCG@5 while keeping other parameters fixed. As we can see in Figure 4.7a, the optimal value of α for Foursquare's is 0.5 and the performance drops for all other values of α . We observe a similar behavior on Gowalla's in Figure 4.7b where the best performance is achieved with $\alpha = 0.9$. For all other values of α , we see a drop in the performance.

4.5.6 Model's Convergence

Finally, Figure 4.8 plots the value of the joint objective function, $\Theta^t = R(U^t, V^t) + R_{\Pi}(U^t, V^t)$ (Eq. (4.4) and (4.9)) when training the model in t iterations/epochs. We observe that the value of Θ^t consistently decreases as the training epochs increase until the proposed JTCR model converges. The behavior of the proposed

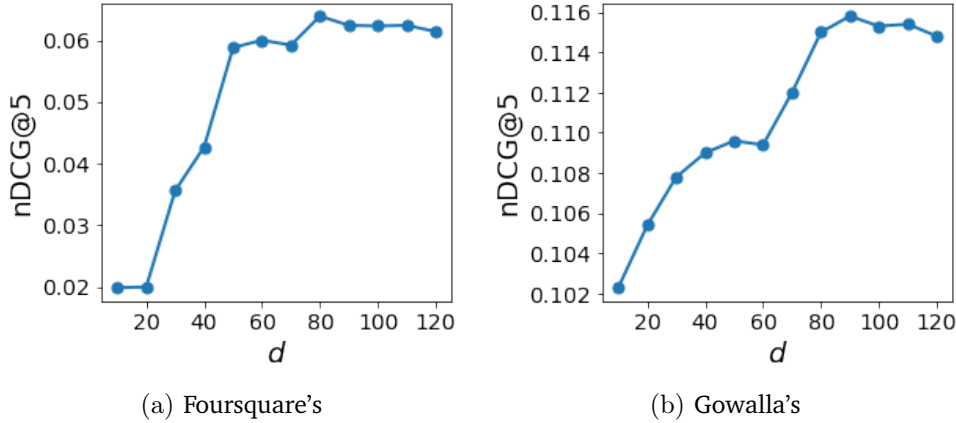


Figure 4.5. Impact of the number of latent factors.

objective function is as expected, since it is the summation of a logistic loss and a quasi convex function. R , that is a logistic loss, it is convex and monotonic decreasing. Also, R_{Π} is strictly positive and monotonic decreasing. Hence, the summation of both loss functions converges, as illustrated in Figure 4.8.

4.6 Summary

In this chapter, we presented an extensive data analysis on two POI recommendation datasets studying various attributes related to sparsity, time-sensitivity, and multiple check-ins. Based on the intuitions we got from data analysis, we proposed a two-phase CR model, called JTTCR. In addition, we showed how to incorporate the geographical influence into the objective functions and proposed a time-sensitive regularizer to capture the long-term user behavior and POI popularity patterns. The experimental results on the two benchmark datasets demonstrated that our proposed model outperforms other state-of-the-art methods. The results indicated that our model is able to address the data sparsity problem taking into account both visited and unvisited POIs in the training phase and their respective geographical distances. We also showed that the second phase is able to rank more-relevant POIs higher in the ranking, explaining the superiority of our two-phase model over the baselines as well as the first phase of the algorithm. This suggests that while single check-ins provide valuable information about the users' preferences, multiple check-ins give us a more clear picture of their behavior and habits. Therefore, in the first phase of our algorithm, a CR model that focuses on ranking visited venues higher than unvisited ones addresses the data

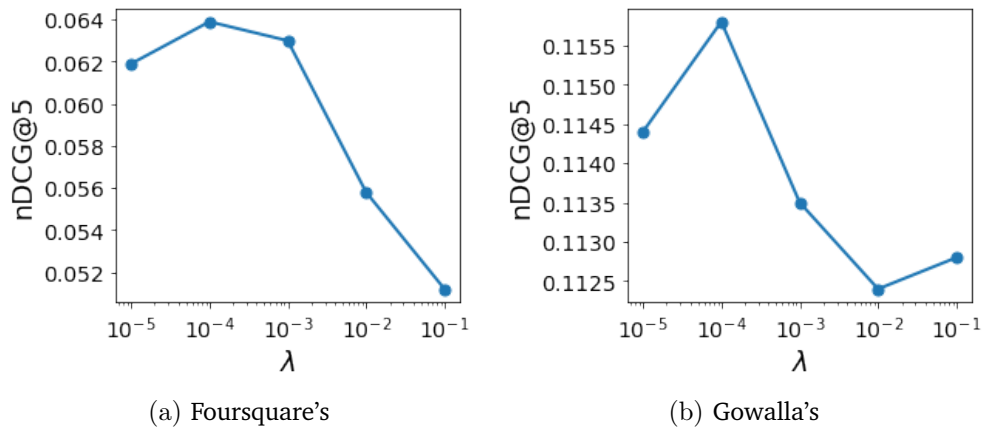


Figure 4.6. Impact of λ .

sparsity by taking into account the unvisited venues in the training phase. In the second phase, a different CR approach is employed focusing on placing users' favorite POIs higher in the ranking. Throughout this process, we have regularized the learning procedure following the intuitions that we had by analyzing time-sensitivity of users and POIs. Our aim was to penalize POIs and users that have been more time-sensitive in the past.

In the next chapter, in an attempt to combine the two approaches we have studied so far, we will propose a hybrid model. From the lessons we learned in Chapters 3 and 4, we model users and POIs using both content-based and collaborative models and fuse the predictions of both models to achieve a higher performance. Our main motivation is to take advantage of both approaches and use the complementary information that these models learn.

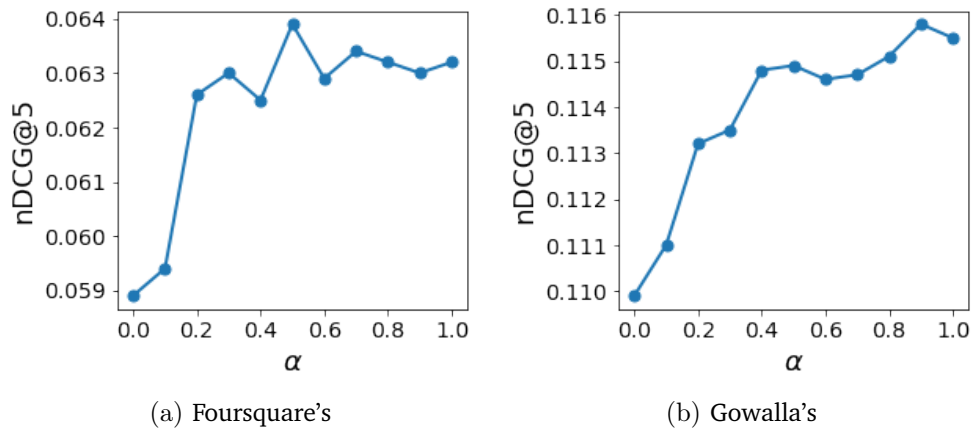
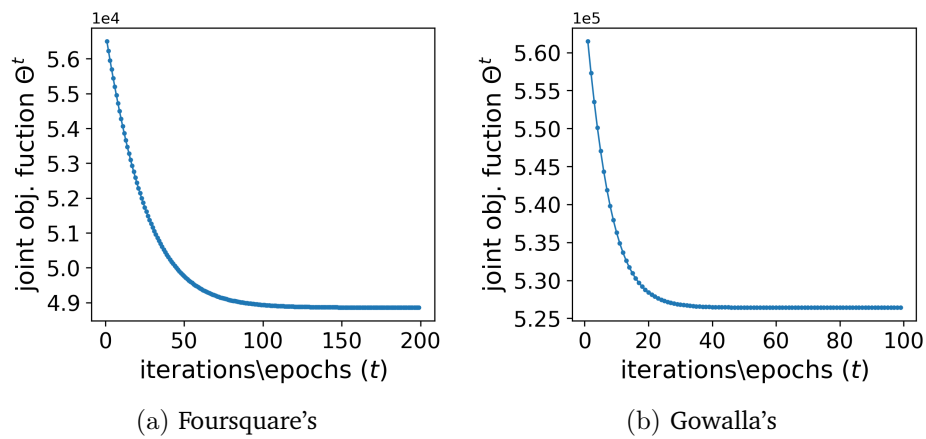
Figure 4.7. Impact of α .

Figure 4.8. Convergence of the joint objective function.

Chapter 5

Hybrid User Modeling for Venue Suggestion

5.1 Introduction

While collaborative approaches are strong in terms of capturing POI and user latent associations on large datasets, they fail to model complex user behavior and preference. Content-based approaches, on the other hand, can effectively model users and POIs individually, but fail to consider latent associations. That is why in the majority of cases, the best performing models are those that combine the two approaches. Such models are called hybrid recommendation models. In this chapter, we aim to explore hybrid recommendation on the TREC-CS dataset. Relevant literature [25] has explored existing content-based, collaborative, and hybrid approaches for this task and has concluded that hybrid models perform the best. As we mentioned earlier, the main reason is that hybrid approaches take advantage of both individual user and POI profiles that are created by a content-based model as well as latent user-POI associations that are learned by a collaborative model.

To this aim, we first propose a novel collaborative venue suggestion framework, called CR-MLS, that enables a model to learn the optimum venue ranking with a focus on the top of the ranked list, while integrating additional information about LBSNs into the model. The basic idea behind our proposed method is that the latent association between two users does not necessarily require them to have visited exactly the same venues in the past. On the other hand, if two users have visited very similar venues, we should still be able to use this information to associate those users with each other. In particular, we design the objective function of our CR model to consider the similarity of venues in the loss func-

tion with a focus on ranking relevant venues at the top of the recommendation list. After proposing our CR method, we introduce three example cross-venue similarity measures, each of which focuses on a different aspect.

In particular, we propose a geographical similarity to incorporate the influence of venues in the same neighborhood. Also, we compute a category-based similarity to take into account venues that provide similar services, like serving similar food. We also calculate a review-based similarity score extracting venues' opinion- and context-based similarity. Note that while we introduce three example similarity functions in this chapter, our proposed framework essentially is not limited to this number of similarity measures. The experimental evaluation shows that considering cross-venue similarities while training the CR model improves the performance beating all CF and CR state-of-the-art methods.

As we mentioned earlier, content-based approaches generally perform better in cases where the data is extremely sparse. Hence, we also compare the performance of our proposed framework against our state-of-the-art content-based approach. As expected, our CR model is unable to outperform our content-based approach. However, we propose a simple yet effective hybrid approach, combining the two models. This model is able to combine the merits of both collaborative and content-based approaches, achieving state-of-the-art recommendation performance.

In summary, this chapter's contributions can be summarized as follows:

- We introduce a novel CR framework, called CR-MLS, with the focus on the top of the recommendation list, while incorporating the cross-venue similarities into the model.
- In order to demonstrate the effectiveness of our CR framework, we propose three different example similarity functions each of which focuses on a different aspect.
- We also propose a simple yet effective hybrid recommendation system, called CR-MLS-Hybrid.

The experimental results on data from the TREC Contextual Suggestion track show that our model alleviates the sparsity problem associating similar venues while training the CR model at different settings.

5.2 Proposed Method

Let $\mathcal{P} = \{\rho_1, \dots, \rho_n\}$ and $\mathcal{L} = \{l_1, \dots, l_m\}$ be the sets of n users and m venues, respectively. We consider user ratings 1, 2, and 3 on venues as negative feedback, while ratings 4 and 5 as positive. For each user ρ_i , we define \mathcal{L}_i^+ as the set of relevant venues, and \mathcal{L}_i^- as the set of irrelevant ones. Moreover, let $S_z \in \mathbb{R}^{m \times m}$ be the similarity matrix of venues based on a similarity feature z .

We aim at computing a personalized ranking function $f_i(l)$ for each user ρ_i to rank relevant venues higher than irrelevant ones. Let $U \in \mathbb{R}^{d \times n}$ be the latent factor of users and $V \in \mathbb{R}^{d \times m}$ be the latent factor of venues, with \mathbf{u}_i and \mathbf{v}_j corresponding to ρ_i and l_j , respectively. For user ρ_i the ranking of the venue l_j is computed as follows $f_i(l_j) = \mathbf{u}_i^T \mathbf{v}_j$. The goal of our model is to learn the latent matrices U and V .

The rest of the section is organized as follows, first we present the CR model that considers cross-venue similarities to generate venue recommendations, and then we introduce the set of examples similarity measures to calculate how close two venues are based on their content and context. Finally, we show how we can combine the ranking of our model with a content-based method resulting in a hybrid model.

5.2.1 Collaborative Ranking with Multiple Location-based Similarities

In this section, we present our Collaborative Ranking framework, called CR-MLS, to suggest venues for each user ρ_i placing relevant venues at the top of the recommendation list. Our goal is to understand the user's check-in behavior in relation to the similarities of venues (see Section 5.2.2). For example, a user may like all venues that are in the city center and serve pizza. Building ranking functions considering different similarities between venues also allows us to model latent associations between users with similar tastes who would not be considered in a traditional CR setting. This happens because CR-MLS takes into account the venue similarities as it updates the user and item latent matrices. CR-MLS can build the associations between users as it considers content- and context-based similarities while updating the latent matrices. Notice that our CR-MLS model does not rely on the type of similarity and is not limited to a certain number of similarity features. Hence, it can be a general framework for incorporating any type of similarity features.

We focus on ranking the venues that a user likes higher than the ones they do not. Formally, we aim at ranking venues that belong to \mathcal{L}_i^+ higher than those

that are in \mathcal{L}_i^- . Our goal is to rank the venues with emphasis on the top of the list. Let $H_i(l_j^-)$ be the height of an irrelevant venue, that is:

$$H_i(l_j^-) = \sum_{k \in \mathcal{L}_i^+} \sum_{z=1}^{|\mathcal{S}|} \left[(\alpha_z \times \mathbf{1}_{[f_i(l_k^+) \leq f_i(l_j^-)]}) / S_z(k, j) \right],$$

where α_z is the weight of similarity S_z and $\mathbf{1}_{[\cdot]}$ is an indicator function. Note that α_z controls the model's bias towards similar venues and can be used to prevent the "Harry Potter" problem [115]. Dividing the indicator function by S_z allows the model to incorporate the similarities into the model while constructing the height for irrelevant items. For example, if an irrelevant item is ranked higher than a relevant item, but they are very similar based on S_z , then the denominator will be higher, which means the height of the irrelevant venue will be reduced proportionally. The objective function should aim at minimizing H_i for all irrelevant venues of user ρ_i . A lower value of H_i means that there are fewer irrelevant venues ranked higher than relevant ones, and those that are ranked higher are more similar to relevant items. However, indicator functions are not convex and they are not suitable to our optimization strategy. Therefore, we use the logistic loss of the difference between the two functions as a convex upper bound surrogate. We define the difference between the k^{th} venue and the j^{th} as follows:

$$\delta_i(k, j) = \mathbf{u}_i^T \sum_{z=1}^{|\mathcal{S}|} \left[\alpha_z (\mathbf{v}_k - \mathbf{v}_j) / \exp(|S_z(k, j)|) \right].$$

Therefore, the surrogate height function $H'_i(l_j^-)$ becomes:

$$H'_i(l_j^-) = \sum_{k \in \mathcal{L}_i^+} \log \left[1 + \exp(-\delta_i(k, j)) \right],$$

where $\log(1 + \exp(-\delta))$ is the logistic loss of δ . Therefore, the objective function of CR-MLS can be reformulated as a minimization problem with respect to the latent matrices U and V as follows:

$$R(U, V) = \sum_{i=1}^m \frac{1}{n_i} \sum_{j \in \mathcal{L}_i^-} (H'_i(l_j^-))^2 = \sum_{i=1}^m \frac{1}{n_i} \times \sum_{j \in \mathcal{L}_i^-} \left(\sum_{k \in \mathcal{L}_i^+} \log \left(1 + \exp \left(-\mathbf{u}_i^T \sum_{z=1}^{|\mathcal{S}|} \left[\alpha_z (\mathbf{v}_k - \mathbf{v}_j) / \exp(|S_z(k, j)|) \right] \right) \right) \right)^2. \quad (5.1)$$

For solving the optimization problem of (5.1), we use a gradient-descent-based alternating optimization algorithm. We first keep V fixed and update U , and then

keep U fixed and update V . Therefore, the update rules of the $t + 1$ iteration are:

$$\mathbf{u}_i^{t+1} = \mathbf{u}_i^t - \gamma \nabla_{\mathbf{u}_i} R(U^t, V^t), \forall i = 1, \dots, n, \quad (5.2)$$

$$\mathbf{v}_j^{t+1} = \mathbf{v}_j^t - \gamma \nabla_{\mathbf{v}_j} R(U^{t+1}, V^t), \forall j = 1, \dots, m. \quad (5.3)$$

For reading simplicity we define

$$\theta(k, j) = (1 + \exp(\delta(k, j))).$$

The gradients of $R(U, V)$ with respect to \mathbf{u}_i and \mathbf{v}_j are computed as follows:

$$\begin{aligned} \nabla_{\mathbf{u}_i} R(U, V) &= \\ &= \frac{2}{n_i} \sum_{j \in \mathcal{L}_i^-} \left(H'_i(l_j^-) \sum_{k \in \mathcal{L}_i^+} \sum_{z=1}^{|\mathcal{S}|} \left[\alpha_z (\mathbf{v}_j - \mathbf{v}_k) / (\exp(|S_z(k, j)|) \theta(k, j)) \right] \right), \end{aligned}$$

$$\begin{aligned} \nabla_{\mathbf{v}_j} R(U, V) &= \\ &= \sum_{i \in \mathcal{P}_j^-} \frac{2}{n_i} \sum_{h \in \mathcal{L}_i^-} \left(H'_i(l_h^-) \sum_{k \in \mathcal{L}_i^+} \sum_{z=1}^{|\mathcal{S}|} \left[\alpha_z \mathbf{u}_i / (\exp(|S_z(k, h)|) \theta(k, h)) \right] \right) \\ &\quad - \sum_{i \in \mathcal{P}_j^+} \frac{2}{n_i} \sum_{h \in \mathcal{L}_i^-} \left(H'_i(l_h^-) \sum_{k \in \mathcal{L}_i^+} \sum_{z=1}^{|\mathcal{S}|} \left[\alpha_z \mathbf{u}_i / (\exp(|S_z(k, h)|) \theta(k, h)) \right] \right), \end{aligned}$$

with \mathcal{P}_j^+ being the set of users who gave a positive rating to l_j and \mathcal{P}_j^- the set of users who gave a negative rating to l_j . Notice that we also consider a regularization term $(\lambda/2)(|U|^2 + |V|^2)$ to avoid model overfitting in our optimization strategy, where λ is the regularization parameter. Then the final venue recommendations are generated by computing the factorized matrix as the product of U and V .

5.2.2 Cross-Venue Similarities

In this section, in order to demonstrate the effectiveness of our proposed framework, we introduce three example similarity measures. We compute similarity measures between two venues l_i and l_j based on their content and location. Let $S_{ij} = \{S_z(i, j) : z \in \{1, 2, 3\}\}$ be the set of similarity functions, which are detailed in the following.

Geographical similarity. First, we compute the geographical similarity between two venues to incorporate the geographical context while characterizing the user's geographical preferences. The similarity is inversely proportional to the distance between two venues. This score is inspired by related studies [125, 126] where a simple geographical measure improved the models significantly. We use the Haversine formula to compute the angular distance between l_i and l_j :

$$\delta_{ij} = 2 \times \arcsin \left(\sqrt{\sin^2(\Delta\phi_{ij}/2) + \cos\phi_i \times \cos\phi_j \times \sin^2(\Delta\eta_{ij}/2)} \right),$$

where ϕ_i and ϕ_j are latitudes of l_i and l_j in radian, respectively. Accordingly, η_i and η_j are longitudes of l_i and l_j in radian. Then we calculate the geographical similarity between l_i and l_j as follows:

$$S_1(i, j) = \frac{1}{1 + (\delta_{ij} \times R)}, \quad (5.4)$$

where R is the earth's radius ($R=6,371\text{KM}$).

Review-based similarity. Online reviews contain a wealth of information about venues as they reflect users' opinions. Since many users explain their context while writing reviews as in, for example: "I had a quick lunch with my friend right after school," it is crucial to measure how similar two venues are in terms of the reviews they received. It is also important to consider how a particular user rated venues that are similarly reviewed by others. Therefore, we train a Support Vector Machine (SVM) classifier with linear kernel to estimate the review-based similarity. The choice of SVM classifier was inspired by observing its notable performance in other studies [6, 14]. For each venue, we train a different SVM classifier. We take the positive reviews of the corresponding venue as positive training samples and the negative reviews as negative training samples. We denote the trained SVM classifier of l_i as SVM_i . Notice that the reviews used for training are independent of a particular user's reviews about a specific venue. In other words, we train the SVM classifiers using the online public reviews available on LBSNs. Finally, we compute the review-based similarity between l_i and l_j by classifying the reviews of l_j using SVM_i . Note that we use both positive and negative reviews of l_j to classify l_j with SVM_i . The value of SVM_i 's decision function computes the similarity of two venues l_i and l_j , denoted as $S_2(i, j)$.

Category-based similarity. While it is essential to exploit users' ratings considering geographical proximity and review-based similarity, it is also crucial to take into account how users rate venues that are similar in terms of their categories. For example, a user who likes pizza is more likely to visit a pizza place and rate

it positively. It has been shown in relevant works that incorporating venue categories into the recommender system is crucial [216]. We calculate the cosine similarity between the vectors of categories associated with venues l_i and l_j on LBSNs as follows:

$$S_3(i, j) = \frac{\mathbf{c}_i \cdot \mathbf{c}_j}{|\mathbf{c}_i|_2 |\mathbf{c}_j|_2}, \quad (5.5)$$

where \mathbf{c}_i and \mathbf{c}_j are the category vectors for l_i and l_j , respectively.

5.2.3 System Overview

Algorithm 2 summarizes our proposed CR-MLS with the three cross-venue similarity measures that we introduced in Section 5.2.2. As we can see from line 1 to 5, the three similarity scores are computed for all the pairs of venues and stored in a three-dimensional matrix called S . Then, from line 7 to 13, the main steps of CR-MLS are done to learn the parameters of the model, taking the similarity of venues into account.

Efficiency. Note that one could argue that computing the similarity measures between all pairs of venues is not efficient. Although this is a valid argument, it is worth noting that the main focus of this thesis is not on efficiency but on effectiveness. However, we believe that a more efficient strategy for selecting venue pairs could be studied to improve the complexity of computing S .

5.2.4 Hybrid Venue Suggestion

In this section, we combine the output of CR-MLS with the output of a state-of-the-art content-based method called LinearRankRev [10]. Our goal is to demonstrate the effectiveness of our approach when combined with a content-based approach on a highly sparse dataset. To this aim, we first produce the ranking using both methods and consider the ordinal position of a venue as its score. For example, the first venue in a ranked list gets the score of 1 and the score of the second one becomes 2. Let $Rk_S(\rho_i, l_j)$ be the ordinal position of venue l_j for user ρ_i using CR-MLS and $Rk_L(\rho_i, l_j)$ be the ordinal position of l_j for ρ_i using LinearRankRev. We calculate the linear combination of the two ranked lists as follows:

$$Rk(\rho_i, l_j) = \beta \times Rk_S(\rho_i, l_j) + (1 - \beta) \times Rk_L(\rho_i, l_j),$$

where β is the combination weight ranging between 0 and 1. The final ranking is obtained by sorting the venues in terms of Rk . In the following section, we call the results of this model CR-MLS-Hybrid.

Algorithm 2: Collaborative Ranking with Multiple Location-based Similarities Algorithm (CR-MLS)

Input: $\mathcal{P}, \mathcal{L}, \maxIter, \mathbf{c}, \phi, \eta, \{d, \lambda, \alpha, \epsilon\}$

Output: U, V

```

1 forall the  $i \in |\mathcal{L}|$  do
2   forall the  $j \in |\mathcal{L}|$  do
3      $S_1(i, j) \leftarrow 1/(1 + (\delta_{ij} \times R))$  (Equation (5.4))
4      $S_2(i, j) \leftarrow$  value of decision function of  $SVM_i$  given reviews of  $l_j$ 
5      $S_3(i, j) \leftarrow (\mathbf{c}_i \cdot \mathbf{c}_j) / (|\mathbf{c}_i|_2 |\mathbf{c}_j|_2)$  (Equation (5.5))
6  $t \leftarrow 0$ 
7 Initialize  $U^{t+1}, V^{t+1}$ 
8  $\theta^{t+1} \leftarrow R(U^{t+1}, V^{t+1}), \theta^t = \theta^{t+1}/2$ 
9 while ( $abs(\theta^{t+1} - \theta^t) > \epsilon$ )  $\wedge$  ( $t < \maxIter$ ) do
10    $t \leftarrow t + 1$ 
11   Update  $\mathbf{u}_i^{t+1}, \forall i = 1, \dots, n$  (Equation (5.2))
12   Update  $\mathbf{v}_j^{t+1}, \forall j = 1, \dots, m$  (Equation (5.3))
13    $\theta^{t+1} \leftarrow R(U^{t+1}, V^{t+1})$ 
14   end
15  $U \leftarrow U^{t+1}, V \leftarrow V^{t+1}$ 

```

5.3 Experimental Setup

In this section, we first introduce the experimental setup describing the dataset, evaluation metrics, parameter tuning as well as compared methods. Then we present the results together with detailed discussions.

5.3.1 Data

We evaluate our approach on a benchmark dataset, made available by the TREC. The dataset is the combination of the data of the TREC-CS 2015 and 2016 tracks [96]. Since TREC released the ground truth for 211 and 58 users in TREC-CS 2015 and 2016, respectively and the settings for both datasets were identical, we combined both datasets to generate a single larger dataset, denoted as *TREC-CS*. In doing so, the sparsity of the user-venue matrix is increased. The task was to produce a ranked list of venues in a new city for users given their history of venue preferences in other cities. Each user has visited and rated 30 to 60 venues in one or two cities. We used the publicly available crawls of [15] as additional information. More specifically, we used additional information from Yelp such as reviews, categories, and address. We then used HERE API¹ to extract geographical coordinates given a venue’s address. In summary, the unified TREC-CS dataset consists of 269 users. The auxiliary information was crawled from Yelp for 6,346 venues. The average number of reviews per venue is 105.55 and the average number of categories per venue is 2.44.

5.3.2 Metrics

We use the official evaluation metrics of TREC for this task, that is, P@k (Precision at k) and nDCG@k with $k \in \{1, 2, 3, 4, 5\}$. Since our approach exploits the influence of neighboring venues, evaluating recommendation in a new city where the user does not have any check-in records does not allow us to study the effect of the geographical similarity function. Hence, we evaluate our method in the same way as the state-of-the-art approaches evaluated their works [125], that is, we use 70% of the check-in data as *training set* (17.9K ratings), 10% as *validation set* (2.5K ratings), and 20% as *testing set* (5.4K ratings). Notice that since the ratings are not timestamped, we split the dataset randomly; hence we repeat our experiments 5 times and report the average P@k and nDCG@k.

¹<https://developer.here.com/>

5.3.3 Compared Methods

We compare our CR-MLS and CR-MLS-Hybrid models with approaches that consider ranking and geographical influence for venue suggestion and approaches based on collaborative ranking with emphasis on the ranking performance at the top of the list. We also compare our models with the TREC’s best performing run. Thus, we compare CR-MLS and CR-MLS-Hybrid with the following methods:

– Collaborative methods:

- **P-Push** [60] focuses on the ranking performance at the top of the list using a p -norm height function in CR. P-Push does not include any contextual information in its learning strategy.
- **RH-Push** [60] is another push CR model based on reverse height, focusing on the ranking performance at the top of the list. RH-Push does not consider any contextual information in its learning strategy either.
- **IRenMF** [131] is based on weighted MF [143] exploiting two levels of geographical neighborhood characteristics: nearest neighboring locations share more similar user preferences, while locations in the same geographical region may share similar user preferences.
- **GeoMF** [126] augments users’ and venues’ latent factors in the factorization model with activity area vectors of users and influence area vectors of venues, respectively.
- **Rank-GeoFM** [125] is the state-of-the-art venue recommendation algorithm. It is a ranking-based MF model that includes the geographical influence of neighboring venues while learning users’ preference rankings for venues.

– Content-based method:

- **LinearCatRev** [10, 12] is the best performing model of TREC-CS 2015 and 2016. It is a content-based recommender system which extracts information from different LBSNs and uses it to calculate category-based and review-based scores. Then, it combines the scores using linear interpolation. The main difference between LinearCatRev and other methods is that it is a content-based method focusing on creating rich user and venue profiles. Although this method performs very well on this dataset, there are major concerns regarding its scalability, mainly because it trains a separate classifier per user, something that can be challenging in a real-life recommendation scenario.

We compare the performance of our proposed models with these methods in the following section.

5.4 Results and Discussion

In this section, we present the results of our proposed model compared against state-of-the-art recommendation methods. Furthermore, we study the impact of the model parameters and components.

5.4.1 Performance Comparison

Tables 5.1 and 5.2 report the performance of all the models on TREC-CS in terms of $P@k$ and $nDCG@k$ with $k \in \{1, 2, 3, 4, 5\}$, respectively. We observe that the push CR-based models perform worse in terms of $P@5$ and $nDCG@5$. This occurs because the baseline push CR-models do not consider any similarities while training the model. Although P-Push performs more effectively than RH-Push regarding $P@1-4$, it has the worst performance in terms of $P@5$ among all models because P-Push focuses on optimizing the model for the negative items, while RH-Push focuses on the positive items. However, since none of them take into account the similarities between venues, they cannot perform as well as the other models. Among the methods that consider geographical influence in the model, Rank-GeoFM performs better. This is because Rank-GeoFM considers venue suggestion problem as a ranking problem, similar to our approach. Rank-GeoFM, IRenMF, and GeoMF perform better than CR-based baselines indicating that geographical influence is an important factor in venue suggestion. While Rank-GeoFM and GeoMF perform similarly in terms of $P@5$, we observe that Rank-GeoFM performs better in term of $nDCG@5$ indicating that a ranking-based approach enables a system to rank more relevant items higher in the ranking.

Our proposed CR-MLS model significantly outperforms all collaborative state-of-the-art methods in terms of $P@1-3$ and $nDCG@1-5$ (according to pairwise t-test at $p < 0.05$). Compared to the state-of-the-art method, Rank-GeoFM, the improvements in terms $nDCG@1$ and $nDCG@5$ are 17% and 11%, respectively. This indicates that our proposed CR-MLS can address the data sparsity problem by incorporating different types of similarities. While the geographical similarity includes the neighborhood influences in the model, the category-based similarity takes into account users with similar tastes when they do not share the same check-in records. In addition to that, the review-based similarity models venues similarities in terms of other users' opinions in various contexts. Fusing these

similarity measures with a CR-based model enables CR-MLS to elaborate complicated similarity affinities among venues and propagate them to the users. Hence, our proposed CR-MLS addresses the data sparsity problem more effectively than other state-of-the-art models, indicated by the high recommendation accuracy. Finally, more improvements in terms of $nDCG@k$ suggests that CR-MLS is able to rank higher venues that are rated higher by the users.

We also compare the performance of our model when combined with a content-based model as a hybrid method, called CR-MLS-Hybrid (see Section 5.2.4). We see in Table 5.1 that LinearCatRev performs better than all collaborative approaches. This result is inline with the findings of Arampatzis and Kalamatianos [25], that is, due to high sparsity of TREC-CS dataset content-based approaches are generally more effective than collaborative methods. However, we observe that CR-MLS exhibits a better performance in terms of $nDCG@1$ compared to LinearCatRev. This motivated us to combine the ranking of CR-MLS and LinearCatRev to build a stronger hybrid approach. As we can see in Table 5.1, CR-MLS-Hybrid outperforms all other methods. In particular, we observe more improvements in terms of $nDCG@k$, indicating that CR-MLS-Hybrid is also able to rank higher the venues with higher rating. It is also worth noting that significant improvement from LinearCatRev indicates that CR-MLS not only performs better than state-of-the-art collaborative methods, but also is able to exploit user-venue associations in a way that the content-based approach fails to do.

5.4.2 Impact of the Number of Visited POIs

Table 5.3 shows $P@5$ and $nDCG@5$ of all models when varying the number of venues that each user has visited in the training set. Table 5.3 shows that CR-MLS achieves the highest accuracy, compared to the other collaborative models and CR-MLS-Hybrid compared to all other models for all different number of venues. This result indicates that CR-MLS can address the sparsity problem better when the training set is smaller. Also, we observe a more robust behavior of CR-MLS compared to the baselines suggesting that incorporating similarities enables the model to deal with noise and data sparsity more effectively. This is more obvious when observing that CR-MLS outperforms all the baseline methods with a larger margin in terms of $nDCG@5$. Also, we can see that LinearCatRev’s performance is less robust as we vary the number of venues. We do not observe the same behavior in CR-MLS-Hybrid’s performance implying that combining the ranking of CR-MLS with LinearCatRev also improves the stability of the content-based approach when trained with less venues in the training set.

Table 5.1. Performance evaluation on TREC-CS in terms of P@k with $k \in \{1, 2, 3, 4, 5\}$. Bold values denote the best scores compared with collaborative approaches and the content-based approach separately.

	P@1	P@2	P@3	P@4	P@5
P-Push	0.5635	0.5179	0.5079	0.4772	0.4524
RH-Push	0.4606	0.4547	0.4580	0.4626	0.4567
IRenMF	0.5037	0.4706	0.4767	0.4706	0.4610
GeoMF	0.4743	0.4871	0.4714	0.4789	0.4740
Rank-GeoFM	0.5662	0.5441	0.5392	0.4926	0.4743
CR-MLS	0.6605^{†‡}	0.5830[†]	0.5510[†]	0.5055	0.4804
LinearCatRev	0.6471	0.6452	0.6336	0.6121	0.5868
CR-MLS-Hybrid	0.6801^{†‡}	0.6673^{†‡}	0.6458[†]	0.6140[†]	0.5919[†]

The superscript [†] denotes significant improvements compared to all collaborative baselines and [‡] denotes significant improvements compared to the content-based baseline (i.e., LinearRankRev), for $p < 0.05$ in paired t-test.

Table 5.2. Performance evaluation on TREC-CS in terms of nDCG@k with $k \in \{1, 2, 3, 4, 5\}$. Bold values denote the best scores compared with collaborative approaches and the content-based approach separately.

	nDCG@1	nDCG@2	nDCG@3	nDCG@4	nDCG@5
P-Push	0.5635	0.5282	0.5188	0.4963	0.4775
RH-Push	0.4606	0.4561	0.4581	0.4611	0.4575
IRenMF	0.5037	0.4781	0.4806	0.4759	0.4689
GeoMF	0.4743	0.4842	0.4879	0.4801	0.4774
Rank-GeoFM	0.5662	0.5491	0.5445	0.5123	0.4976
CR-MLS	0.6605^{†‡}	0.6043[†]	0.5865[†]	0.5614[†]	0.5509[†]
LinearCatRev	0.6471	0.6498	0.6499	0.6444	0.6394
CR-MLS-Hybrid	0.6801^{†‡}	0.6734^{†‡}	0.6672^{†‡}	0.6562[†]	0.6538^{†‡}

The superscript [†] denotes significant improvements compared to all collaborative baselines and [‡] denotes significant improvements compared to the content-based baseline (i.e., LinearRankRev), for $p < 0.05$ in paired t-test.

Table 5.3. Effect on P@5 and nDCG@5 of different number of venues that users visited as training set.

Number of venues	P@5			nDCG@5		
	40	50	60	40	50	60
P-Push	0.4278	0.4346	0.4466	0.4493	0.4375	0.4744
RH-Push	0.4343	0.4556	0.4574	0.4277	0.4659	0.4606
IRenMF	0.4404	0.4588	0.4588	0.4485	0.4654	0.4636
GeoMF	0.4544	0.4618	0.4640	0.4559	0.4573	0.4636
Rank-GeoFM	0.4551	0.4727	0.4728	0.4576	0.5006	0.5027
CR-MLS	0.4677[†]	0.4732	0.4800	0.4736[†]	0.5302[†]	0.5450[†]
LinearCatRev	0.5706	0.5632	0.5721	0.6246	0.6195	0.6260
CR-MLS-Hybrid	0.5772[†]	0.5787^{†‡}	0.5853^{†‡}	0.6357[†]	0.6353^{†‡}	0.6433^{†‡}

The superscript [†] denotes significant improvements compared to all collaborative baselines and [‡] denotes significant improvements compared to the content-based baseline (i.e., LinearRankRev), for $p < 0.05$ in paired t-test.

5.4.3 Impact of the Similarity Scores

Figure 5.1a shows the performance of CR-MLS when varying α_1 , α_2 , and α_3 , keeping in each run the other two parameters fixed. The best performance is achieved with geographical similarity weight at $\alpha_1 = 0.5$, review-based similarity weight at $\alpha_2 = 0.2$, and category-based similarity weight at $\alpha_3 = 0.3$. From Figure 5.1a, we can also see how much each of the similarity measures contribute to the overall performance. To do so, we take the performance of CR-MLS when the value of each α equals zero. This values indicates the performance of CR-MLS when the respective similarity measure is removed from the model. More specifically, the performance of CR-MLS in terms of nDCG@5 removing each of the similarity scores is as follows:

- CR-MLS: nDCG@5 = 0.5509
- CR-MLS-NoGeographical: nDCG@5 = 0.5288
- CR-MLS-NoReview: nDCG@5 = 0.5375
- CR-MLS-NoCategory: nDCG@5 = 0.5306

We can see that the performance is dropped after removing each of the similarity scores, indicating that each of these similarity scores contribute to the overall performance of CR-MLS. Removing the geographical similarity results in the

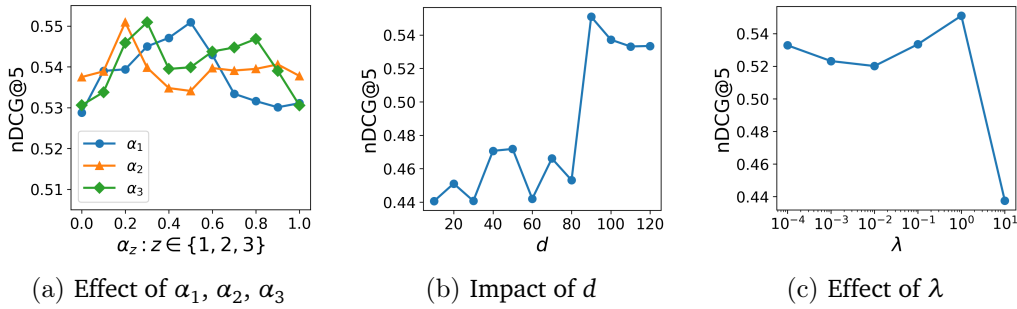


Figure 5.1. Impact of different model parameters on the performance of CR-MLS

highest drop compared to the other similarity measures. This shows that geographical similarity captures the similarity of two venues more effectively and reflects users' preference more accurately.

5.4.4 Impact of the Number of Latent Factors

In Figure 5.1b we study the effect of latent factors d on the performance of our model and report nDCG@5 while keeping other parameters fixed. We observe that the optimal number of latent factors d is 90, and for other values of latent factors, nDCG@5 drops. As we can see, the difference in the performance of the model is more when varying d from 80 to 90. This indicates the importance of finding the optimal number of latent factors for training CR-MLS as it can have large impact on the model's performance since the number of latent factors are crucial while learning the latent associations between users and venues.

5.4.5 Impact of Regularization Parameter

Figure 5.1c shows the effect of the regularizing control parameter (λ). We varied λ while keeping other parameters fixed. We see that CR-MLS performs best with $\lambda = 1$. The performance of CR-MLS drops using different other values of λ .

5.5 Summary

In this chapter, we presented a similarity-aware collaborative ranking framework for venue suggestion, called CR-MLS. The proposed CR-MLS is able to include an arbitrary number of cross-venue similarity measures in the model's objective

function enabling the model to propagate venue affinities to the users and hence address the data sparsity problem. To demonstrate the performance of CR-MLS, we also proposed three example cross-venue similarity measures focusing on different aspects. Geographical similarity incorporates the neighborhood influence of venues while category-based similarity takes into account venues that provide similar services. A review-based similarity score was also computed extracting an opinion- and context-based similarity of venues. We compared the performance of CR-MLS with five collaborative and one content-based state-of-the-art approaches on a combined dataset of two publicly available TREC collections. The results indicated that our model can address the data sparsity problem, outperforming the state-of-the-art methods significantly. While we introduced three example similarity functions, it should be noted that CR-MLS is very flexible to incorporate other features.

Part II

Mobile Search

Chapter 6

Unified Mobile Search

6.1 Introduction

Recent years have witnessed a rapid growth in the use of mobile devices, enabling people to access the Internet in various contexts. More than 77% of Americans now own a smartphone,¹ with an increasing trend in terms of the time people spend on their phones. As of 2016, the average U.S. user spends 5 hours on mobile devices per day, with just 8% of it spent in the phone's browser. In fact, people spend most of their time (72%) using apps that have their own search feature.² Moreover, Google Play Store now features more than 3.5 million apps and users install an average of 35 mobile apps on their phones, using half of them regularly.³

More recently, with the release of intelligent assistants, such as Google Assistant and Apple Siri, people are experiencing mobile search through a single voice-based interface. These systems introduce several research challenges. Given that people spend most of their times in apps and, as a consequence, most of their search interactions would be with apps (rather than a browser), one limitation is that users are unable to use a conversational system to search within many apps. This suggests the need for a *unified search framework* that *replaces all the search boxes in the apps, with a single search box*. The workflow of a unified mobile search framework is presented in Figure 6.1. As we see in this figure, with such a framework, the user can submit a query through this system which will identify the target app(s) for the issued query. The query is then routed to the

¹<http://www.pewinternet.org/fact-sheet/mobile/>

²<http://flurrymobile.tumblr.com/post/157921590345/>

³<https://www.thinkwithgoogle.com/advertising-channels/apps/app-marketing-trends-mobile-landscape/>

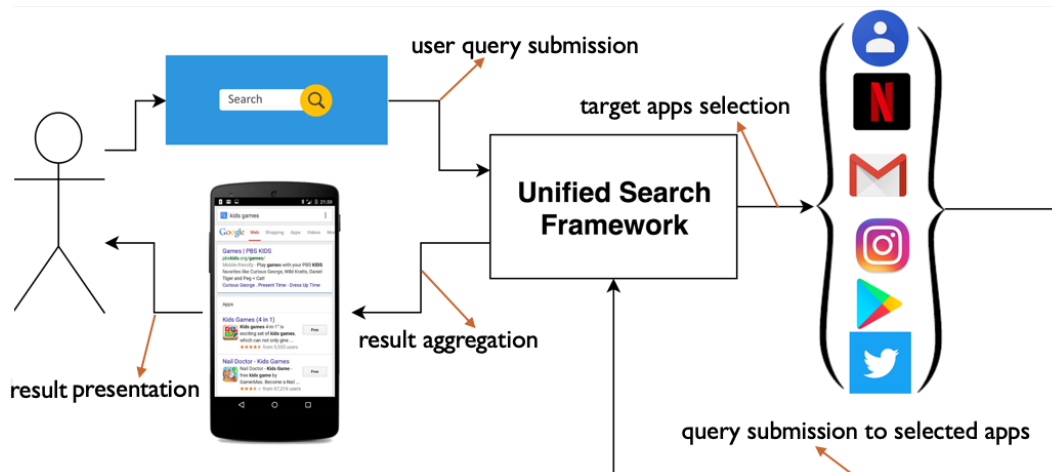


Figure 6.1. Workflow of an example unified mobile search framework.

identified target apps and the results are displayed in a unified interface.

After a thorough investigation and modeling of user behavior for POI recommendation in previous chapters, here we study user behavior in an environment where users express their information needs explicitly via queries. Even though research on mobile IR has been going on for over a decade [73], only now researchers have studied app-based user behaviour. Moreover, the recent advances of mobile apps, social media, and conversational agents increase the need for analysis and modeling of user behavior and information need for unified mobile search. In this chapter, we are particularly interested in taking the first step towards developing a unified search framework for mobile devices by introducing and studying the task of *target apps selection*, which is defined as identifying the target app(s) for a given query. To this end, we built a collection of cross-app search queries through crowdsourcing, which is released for research purposes.⁴ Our crowdsourcing experiment consists of two parts: we initially asked crowdworkers to explain their latest search experience on their smartphones and used them to define various realistic mobile search tasks. Then, we asked another set of workers to select the apps they would choose to complete the tasks as well as the query they would submit. We investigate various aspects of user behaviors while completing a search task. For instance, we show that users choose to complete most of the search tasks using two apps. In addition, we demonstrate that for the majority of the search tasks, most of the users prefer *not* to use Google Search.

⁴Available at <http://aliannejadi.github.io/unimobile.html>

From the lessons learned from our data analysis, we propose two simple yet efficient neural target apps selection models. Our first model looks at the problem as a ranking task and produces a score for a given query-app pair. We study two different training settings for this model. Our second framework, on the other hand, casts the problem as a multi-label classification task. Both neural approaches, called NTAS, learn a high-dimensional representation for each app. Our experiments demonstrate that our model significantly outperforms a set of state-of-the-art models in this task.

In summary, the main contributions of this chapter include:

- Designing and conducting two crowdsourcing tasks for collecting cross-app search queries for real-life search tasks. The tasks and queries are publicly available for research purposes.
- Presenting the first study of user behaviors while searching with different apps as well as their search queries. In particular, we study the attributes of the search queries that are submitted to different apps and user behaviors in terms of the apps they chose to complete a search task.
- Proposing two neural models for target apps selection.
- Evaluating the performance of state-of-the-art retrieval models for this task and comparing them against the proposed method.

Our analyses and experiments show the good performance of our proposed model and suggest specific future directions in this research area.

6.2 Data Collection

In this section, we describe how we collected *UniMobile*, which is, to the best of our knowledge, the first dataset on cross-app mobile search queries. We started by creating a number of Human Intelligence Tasks (HITs) on Amazon Mechanical Turk,⁵ asking workers to describe their latest mobile search experience in detail. The answers helped us to define fine-grained diverse naturalistic mobile search tasks. Then, we launched another task asking workers to assume they wanted to complete a given search task on their smartphones. They had to submit their search queries as well as select the apps they would choose to complete each task.

⁵<http://www.mturk.com>

Table 6.1. Distribution of crowdsourcing search task categories.

Search Category	% of tasks
General Information & News	13%
Video & Music	12%
Image	9%
Social Networking	9%
App	9%
File & Contact	8%
Online Shopping	13%
Local Services & Navigation	15%
Email & Event	12%

Task definition. In the first crowdsourcing task, we described the category of search, giving the workers a handful of general examples. Furthermore, we also asked them to give us the context and background of their search, as well as the queries and the apps they used to do the search. Finally, we provided a complete example of a valid answer. We launched this job for most of search categories listed in Table 6.1. The HIT payment was \$0.10 and the workers were based in the U.S. with an overall acceptance rate of 75% or higher. The average work time was 246 seconds with 135 workers completing 169 HITs resulting in an average of 92 terms per HIT. The workers provided enough details about the context and background of their search that enabled us to generalize the task to the level that we would get a wide range of queries on the same task. For example, one worker submitted the following answer:

“I was searching for a new refrigerator to buy. The first thing I did was search for the best refrigerators of 2017 and then narrow down my search for exactly the type of refrigerator that I was looking for...”

Then, we used this answer to define a more general search task:

“Consider one of the oldest appliances in your home. You have been thinking of changing it for a while. Now, it’s time to order it online.”

Query and app pairs. The second crowdsourcing task consisted of 206 individual search task descriptions, mostly extracted from the answers we got in the first task. Table 6.1 lists the distribution of the tasks. In the definition of tasks, our aim was to cover various aspects of mobile information seeking as mentioned

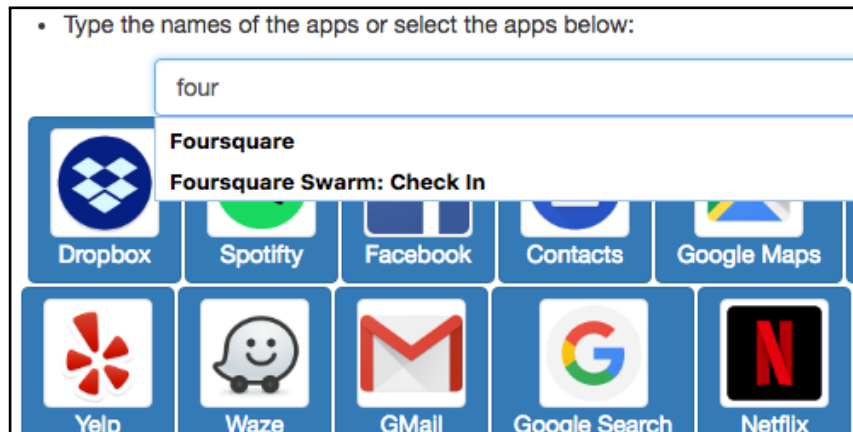


Figure 6.2. HIT interface for choosing apps. The workers could enter an app's name or click on an app's icon.

in [63]. We asked the workers to read the search task description very carefully and assume that they wanted to perform it using their own mobile device. Then, we asked them to select one or more apps from a given list. Alternatively, they could type the name of the app they would choose for that search task. We provided an auto-complete feature for entering the apps' names in order to make it easier for the users to type the name of their favorite apps. Figure 6.2 shows the interface we designed for this HIT. Since we restricted the HIT to be done only by workers in the U.S., we chose the list of apps from the most popular Android apps in the U.S. market. Note that the apps were randomly shuffled and displayed to each worker to prevent any position bias. These apps are listed as follows: Google Search, Gmail, Play Store, Facebook, Instagram, Google Maps, YouTube, Amazon, Twitter, Spotify, Waze, Pinterest, WhatsApp, File Manager, Netflix, Yelp, Contacts, Dropbox.

As incentive, we paid \$0.05 for every HIT assignment. We also encouraged the workers to complete a survey for a \$0.05 bonus. Our aim was to understand the workers' background and familiarity with mobile devices. We asked the workers to perform the task using their mobile devices' browsers and tracked their keyboard keystrokes to prevent them from copying any text from the task description. The average work time for this task was 85 seconds with 91% of the workers completing the survey. The key statistics of the survey were that 59% of the workers used Android and 55% used a mobile device as the primary device to connect to the Internet. Moreover, 83% of the workers believed they use their mobile device more than two hours a day and 41%, more than four hours a day. After launching several batches, we went through all the submitted answers for

Table 6.2. Statistics of UniMobile.

# queries	5,812
# unique queries	5,567
# users	625
# search tasks	206
# unique apps	121
# unique first apps	70
# unique second apps	89
Mean unique apps per task	7.51 ± 10.57
Mean query per user	9.30 ± 20.30
Mean query per task	28.21 ± 12.72
Mean query terms	4.21 ± 2.45
Mean query characters	24.83 ± 12.88

quality control and we observed that following crowdsourcing task design guidelines of [113] helped us achieve a very high assignment approval rate (99%). We have made the collection publicly available for research purposes. The released data consists of the tasks that we defined through the first set of HITs as well as user queries in the second set of HITs, together with their corresponding ranked list of apps. The data can be used to study how users are engaged in searching with different apps. Also, the release of the defined tasks provides the opportunity to conduct a similar study in a lab setting on participants' mobile phones and compare the findings with our results.

6.3 Data Analysis

In this section, we present a thorough analysis of UniMobile, to understand how users issue queries in different apps, and which apps they choose to complete search tasks. With the definition of 206 mobile search tasks, we were able to collect 5,812 search queries and their target apps. Overall, queries were assigned to 121 unique apps. Table 6.2 lists all the details of our dataset. In the following, we analyze different aspects of the data.

6.3.1 App Distribution

Figure 6.3 shows the distribution of queries with respect to users and apps in UniMobile. As we can see in Figures 6.3a and 6.3c, while there exist 173 users

who submitted only one query, 110 users account for 80% of the queries and 239 users account for 95% of the queries. Also, we see in Figures 6.3b and 6.3d that the distribution of apps follows a power-law distribution. In particular, 9 apps account for more than 80% and 17 apps account for more than 95% of the queries. Figure 6.4 shows how queries are distributed with respect to the top 17 apps. As we can see, while Google Search⁶, that is mainly targeted for Web search, constitute 39% of total app selections, users opt to perform the majority (61%) of their search tasks using other apps. Moreover, the variety of apps ranges from apps dealing with local phone data (e.g., Contacts and Calendar) to social media apps (e.g., Facebook and Twitter) indicating that they cover a wide range of search tasks.

App selection. Here we are interested in finding out how users behave while choosing an app to perform a search task. Although users assign two apps while submitting 72% of the queries, they choose only one app for 21% of the queries and choose more than two apps for only 7% of the queries. We also analyze how many different apps users select while doing the tasks. Figure 6.5a shows the distribution of unique apps per users illustrating how many users selected a certain number of different apps. As we can see, a quarter of users preferred to search using two unique apps. On the other hand, Figure 6.5b plots the same distribution with respect to the tasks, that is how many unique apps were selected for each task. We see an entirely different distribution where the average number of unique apps per task is 7.51, showing that the given search tasks can be addressed using multiple apps. As we compare the two distributions in Figures 6.5a and 6.5b, we can conclude that while the majority of search tasks can be addressed using multiple apps, users usually limit their choice to a personal selection of apps. Therefore, a system can define a set of candidate apps which then can be narrowed down considering user's personal preference.

Furthermore, we analyze users' choice of Google Search, observing that it is selected as the first app in 39% of the queries while 46% as the second app. The users chose Google Search as the third app in 30% of the queries with three selected apps. This indicates that, according to UniMobile, in most cases (61%), users prefer to open a more specific app than Web search apps such as Google Search. We also analyze users collective app selection behavior with respect to the tasks. For each task, we count how often each app is selected and sorted them. Our aim is to find out how often users decide to perform their search tasks using Google Search. According to our study, in 14% of the tasks, no user selected Google Search, while in 35% of the tasks Google Search was the

⁶The app "Google Search" is also used to refer to the Google Chrome app.

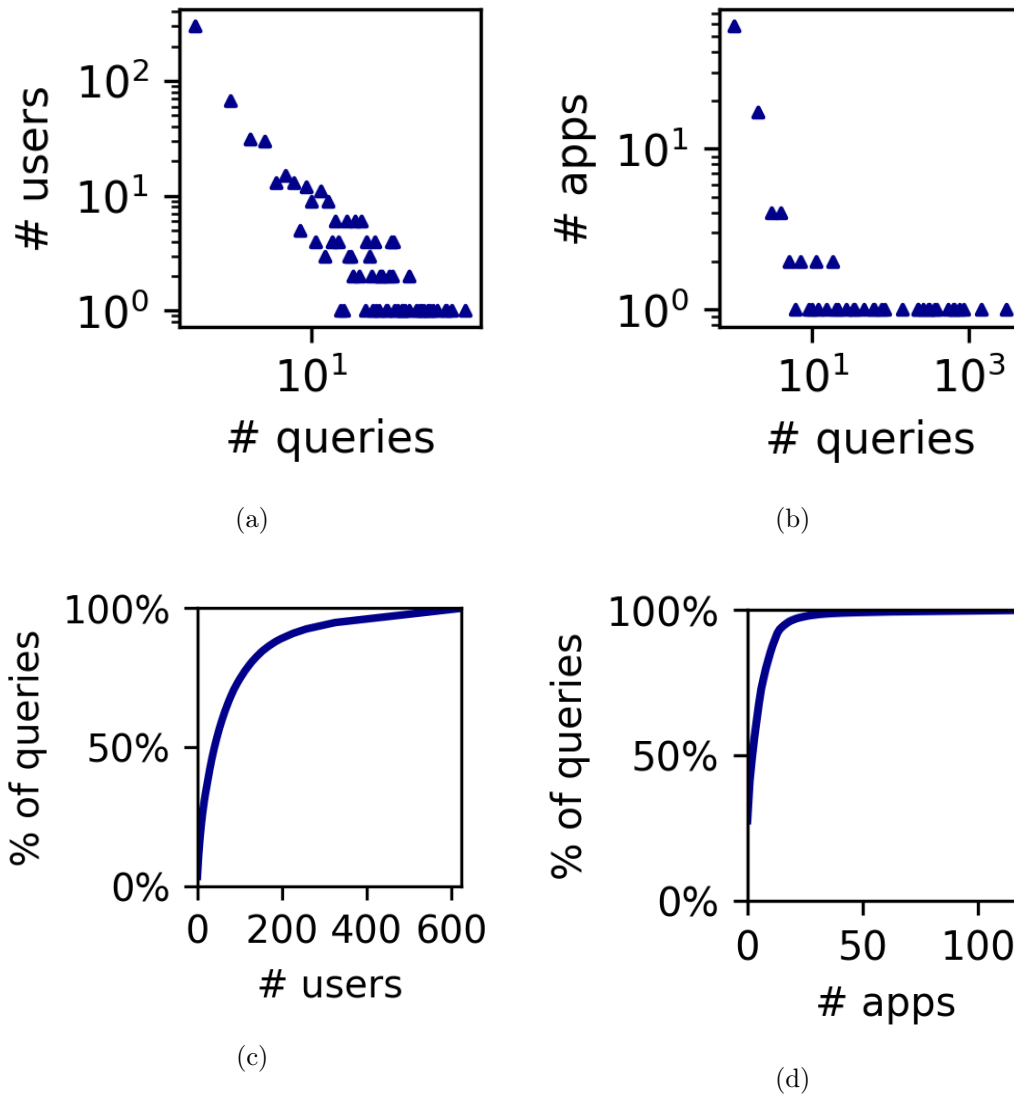


Figure 6.3. The distribution of number of queries with respect to apps and users.

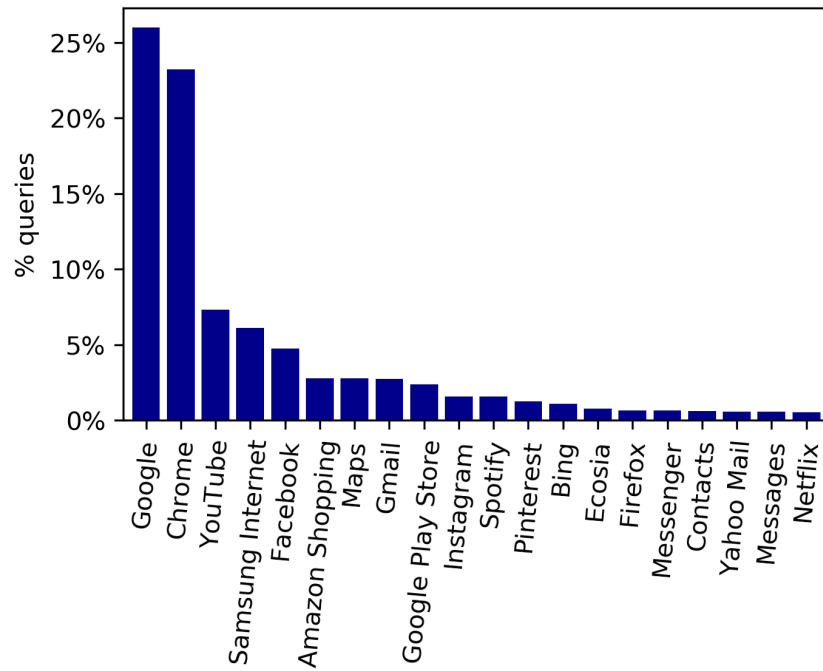


Figure 6.4. Number of queries per app for the top 17 apps.

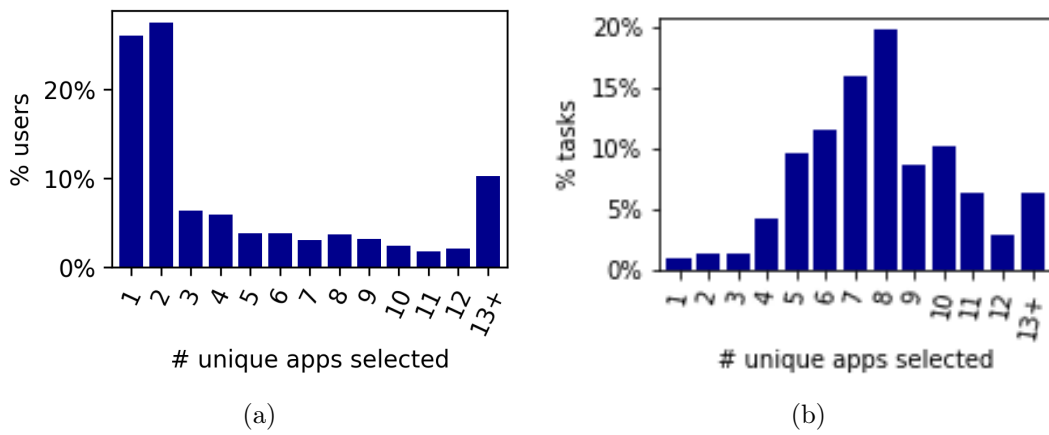


Figure 6.5. Distribution of unique apps per user and task.

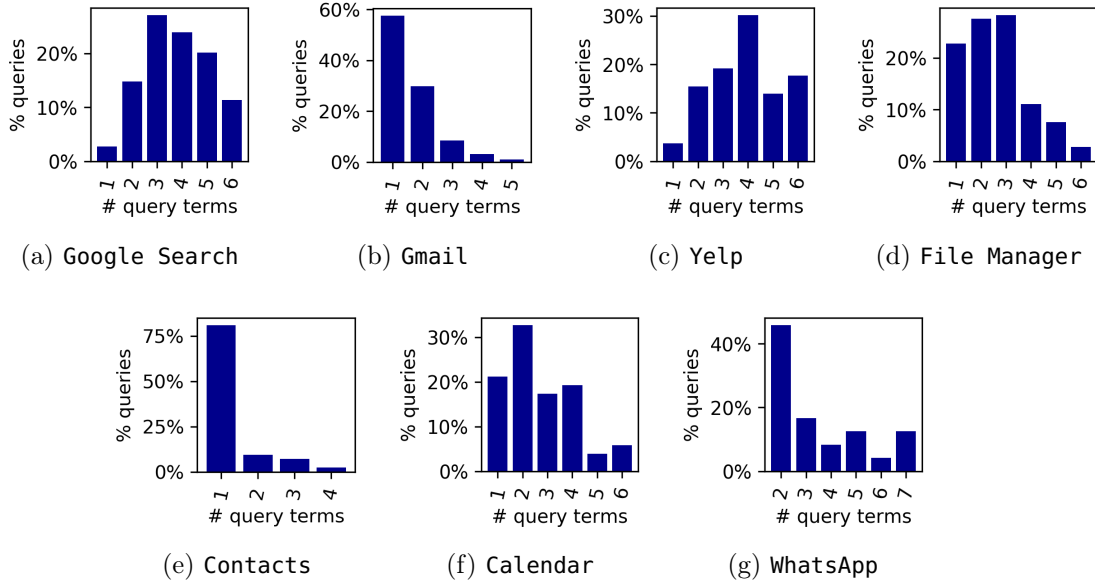


Figure 6.6. Histogram of number of query terms per app.

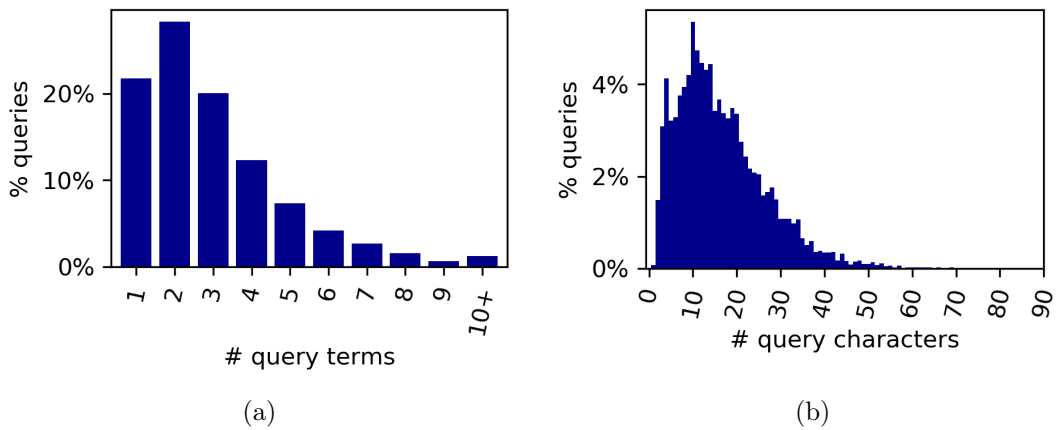


Figure 6.7. Query length distribution with respect to number of terms and characters.

most selected app. Moreover, in 68% of the tasks it was among the top two most frequently selected apps, and in 78% of the tasks it was among the top three. Considering the categorical distribution of apps in Table 6.1 where only 13% of the tasks were in the category of General Information & News, we see that Google Search attracts many queries from the tasks that can be done using a more specific app. Given the integrity and aggregation of various search services such as image, video, location, and online shopping and easy access to them in one app, this observation is not surprising. Nevertheless, we see that for 86% of the tasks, most users preferred other apps. This suggests that a unified mobile search system has a high potential of simplifying and improving users search experience.

6.3.2 Query Attributes

We analyze different attributes of queries with respect to their corresponding apps, to understand how different users formulate their information needs into queries using different apps. After tokenizing the queries, the average query terms per query is 4.21. We analyze the distribution of number of query terms per app, observing different distributions for every app, some of which are shown in Figure 6.6. This difference is more obvious if we compare Google Search with personal or local apps such as Contacts. In particular, Google Search has an average of 4.82 query terms while Contacts has an average of 2.67, which is considerably less than other apps. This can be explained if we consider the type of information users usually look for using the Contacts app. The queries usually consist of one of the stored names on the phone, followed by terms such as “email,” “address,” “info,” and “contact.” Moreover, Figure 6.7 plots the distribution of query length with respect to terms and characters on the whole dataset.

Figure 6.6 demonstrates the distribution of number of query terms for 7 apps. In this figure, we only include the apps that exhibit a considerably different distribution from the average. As shown, Google Search query terms peak at 3 while personal apps such as Contacts, Calendar, and Gmail peak at 2. This indicates that the structure of queries vary depending on the target app. We can also see the difference in the most frequent unigrams for two example apps in Figure 6.8 where we see that while stopwords are the most frequent unigrams used in queries submitted to Google Search, for a specific personal app such as Calendar, a domain-specific term such as “meeting” accounts for more than 15% of the total distribution. This suggests that while considering domain-specific terms is crucial to predicting the target app, taking into account the query structure is also important. For instance, as we see in Figure 6.8a, the question mark

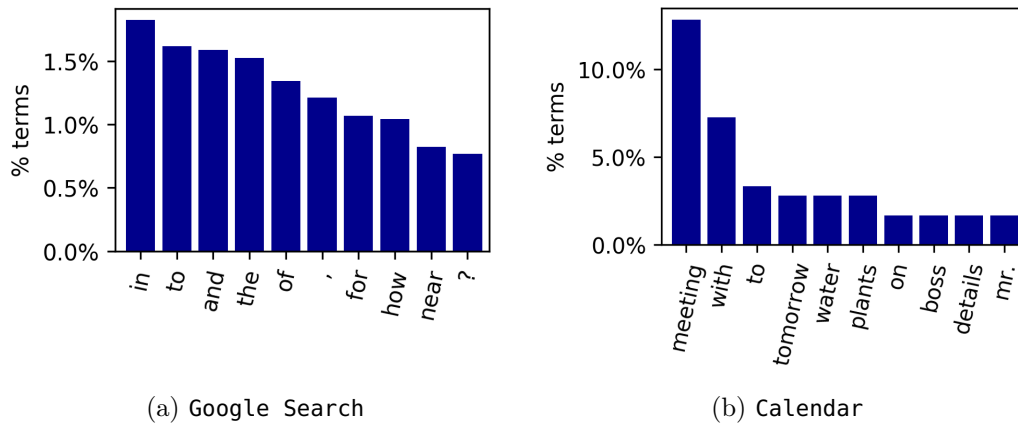


Figure 6.8. Distribution of top query unigrams for two sample apps.

is among the top query unigrams submitted to Google Search, suggesting that many of the queries are submitted in the form of a question. In contrast, as we mentioned earlier, the structure of contact queries are mostly in the form of “<proper noun> + <information field>,” as in “sam email.”

6.3.3 Query Overlap

Here we study query overlap or query similarity over the queries using a simple function used in previous studies done on large-scale query logs (e.g., [64]). We measure the query overlap at various degrees and use the similarity function $\text{sim}(q_1, q_2) = |q_1 \cap q_2| / |q_1 \cup q_2|$. This function simply measures the overlap of query terms. We observed 70% of queries overlapping with at least another query at the similarity threshold of > 0.25 . Higher thresholds lead to significantly lower similar queries; with thresholds > 0.50 and > 0.75 we observe that 24% and 9% of queries were similar, respectively. Similar to previous analyses, in Table 6.3 we observe a different level of query overlap in queries associated with different apps. The least query overlap is observed for Facebook queries. This could be due to the personal environment of Facebook. The highest query overlap is observed in Play Store queries. We observed the presence of some domain-specific terms such as “app” in many queries which results in higher query similarity. The observed difference in query overlap for every app suggests that various factors influence the way users formulate their queries. For example, apps that provide more focused information, receive more similar queries. On the other hand, more personal apps receive a diverse set of queries as they

reflect personal information needs which can be totally different from one user to the other.

6.3.4 Remarks

Our analyses first showed that users' queries are mainly targeted to a few apps; however, these apps are very different in terms of their content. Moreover, we showed that users often choose two different apps for a single query, suggesting that many users submit the same query in multiple apps. Also, we showed that different users select an average of more than 7 apps for each task, with Google Search being the top selected app in only 35% of the cases. This again indicates the necessity of a unified search system on mobile devices. Finally, we analyzed the queries issued in different apps and found notable differences. For instance, we showed that query lengths, unigram distribution, and query overlap differ among apps. This suggests that the query structure needs to be taken into account while representing the apps.

6.4 Neural Target Apps Selection

Assume that a user aims at submitting a query q to a set of mobile apps $\{a_1, a_2, \dots, a_n\}$, called the target apps. Note that the size of this set could be equal to 1. The task of *target apps selection* is defined as ranking the mobile apps in response to the query q , such that the target apps appear in higher ranks. In this section, we propose our methodology to tackle the target apps selection task. To this end, we propose two *general* frameworks based on neural networks. Our first framework, called NTAS1, is given a query and a candidate app and produces a retrieval score. We study both pointwise and pairwise training settings for this framework. Our second framework, called NTAS2, is given a query as the input and produces a probability distribution indicating the probability of each app being targeted, for all apps.

One of the main challenges in this task is that it is not obvious how to represent each app. For example, although the apps' descriptions would be used for app representation in the app selection task [144], it cannot be used in the target apps selection. Because the queries that can be searched in a specific app do not match with the content of the app's description. To address this issue, our frameworks learn a high-dimensional representation for each app, as part of the network. The following subsections describe these two frameworks in more detail.

Table 6.3. The percentage of similar queries at different similarity thresholds considering only the queries associated with every app.

App	% of similar queries		
	> 0.25	> 0.50	> 0.75
All apps	70%	24%	9%
Google Search	63%	19%	6%
Amazon	38%	8%	3%
Gmail	57%	14%	7%
YouTube	49%	20%	7%
Google Maps	46%	3%	1%
Facebook	30%	9%	1%
Play Store	61%	26%	14%

6.4.1 NTAS1: App Scoring Model

NTAS1 outputs a retrieval score for a given query q and a candidate app a . Formally, NTAS1 can be defined as follows:

$$\text{score} = \psi(\phi_Q(q), \phi_A(a)),$$

where $\psi(\cdot, \cdot) \in \mathbb{R}$ is a scoring function for the given query representation $\phi_Q(q) \in \mathbb{R}^m$ and app representation $\phi_A(a) \in \mathbb{R}^n$. Various neural architectures can be employed to model each of the three components in the NTAS1 framework.

We implement the component $\phi_Q(q)$ with two major functions: an embedding function $\mathcal{E} : V \rightarrow \mathbb{R}^d$ that maps each vocabulary term to a d -dimensional embedding space, and a global term weighting function $\mathcal{W} : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The query representation function ϕ_Q represents a query $q = \{w_1, w_2, \dots, w_{|q|}\}$ as follows:

$$\phi_Q(q) = \sum_{i=1}^{|q|} \widehat{\mathcal{W}}(w_i) \cdot \mathcal{E}(w_i), \quad (6.1)$$

which is the weighted element-wise summation over the terms' embedding vectors (hence, $m = d$). $\widehat{\mathcal{W}}$ is the normalized global weights computed using a softmax function as follows:

$$\widehat{\mathcal{W}}(w_i) = \frac{\exp(\mathcal{W}(w_i))}{\sum_{j=1}^{|q|} \exp(\mathcal{W}(w_j))}.$$

This is a simple yet effective approach for query representation based on the bag of words assumption, which has been proven to be effective for the ad-hoc retrieval task [77]. Note that the matrices \mathcal{E} and \mathcal{W} are the network parameters in our model and are learned to provide task-specific representations.

The app representation component ϕ_A is simply implemented as a look-up table. In other words, our neural model consists of an app representation matrix $\mathcal{A} \in \mathbb{R}^{N \times n}$ where N denotes the total number of apps and the i^{th} row of this matrix is a n -dimensional representation for the i^{th} app. Therefore, $\phi_A(a)$ returns a row of the matrix \mathcal{A} that corresponds to the app a .

To model the function ψ , following Zamani et al. [213], we feed the Hadamard product (which enforces $m = n$) of the learned query and app representations into a fully-connected feed-forward network with two hidden layers. This network produces a single output as the score assigned to the given query-app pair. We use rectified linear unit (ReLU) as the activation function in the hidden layers of the network. To prevent overfitting, the dropout technique [179] is employed.

We study both pointwise and pairwise learning settings for our NTAS1 model.

Pointwise learning. In a pointwise setting, we use mean squared error (MSE) as the loss function. MSE for a mini-batch b is defined as follows:

$$\mathcal{L}_{MSE}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} (y_i - \psi(\phi_Q(q_i), \phi_A(a_i)))^2,$$

where q_i , a_i , and y_i denote the query, the candidate app, and the label in the i^{th} training instance of the mini-batch. For this training setting, we use a linear activation for the output layer.

Pairwise learning. NTAS1 can be also trained using a pairwise setting. Therefore, each training instance consists of a query, a target app, and a non-target app. To this end, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [123]. Hinge loss for a mini-batch b is defined as follows:

$$\mathcal{L}_{Hinge}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} \max \{0, \epsilon - \text{sign}(y_{i1} - y_{i2}) (\psi(\phi_Q(q_i), \phi_A(a_{i1})) - \psi(\phi_Q(q_i), \phi_A(a_{i2})))\},$$

where ϵ is a hyper-parameter determining the margin of hinge loss, a linear loss function that penalizes examples violating the margin constraint. To bound the output of the model to the $[-1, 1]$ interval, we use tanh as the activation function for the output layer, in the pairwise training setting. The parameter ϵ is also set to 1, which works well when the predicted scores are in the $[-1, 1]$ interval.

6.4.2 NTAS2: Query Classification Model

Unlike NTAS1 that predicts a score for a given query-app pair, our second framework computes the probability of each app being targeted by a given query. In more detail, NTAS2 is modeled as $\gamma(\phi_Q(q)) \in \mathbb{R}^N$, whose i^{th} element denotes the probability of the i^{th} app being targeted, given the query representation $\phi_Q(q)$. N is the total number of apps.

To implement NTAS2, we represent each query via a weighted element-wise average as explained in Equation (6.1). γ is modeled using a fully-connected feed-forward network with the output dimension of N . ReLU is employed as the activation function in the hidden layers, and a softmax function is applied on the output layer to compute the probability of each app being targeted by the query.

To train NTAS2, we use a cross-entropy loss function which for a mini-batch b is defined as:

$$\mathcal{L}_{ce}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} \sum_{j=1}^N (p(a_j|q_i) \log \gamma(\phi_Q(q_i))) .$$

Similar to NTAS1, we use dropout to regularize the model.

6.5 Experimental Setup

6.5.1 Data

We evaluated the performance of our proposed models on the UniMobile dataset. We followed two different strategies to split the data: 1. In *UniMobile-Q*, we randomly selected 70% of the *queries* for training, 10% for validation, and 20% for test set 2. In *UniMobile-T*, we randomly split the tasks (rather than queries). To do so, we randomly selected 70% of the *tasks* for training, 10% for validation, and 20% for test set. To minimize random bias, for each splitting strategy we repeated the process five times. The hyper-parameters of the models were tuned based on the results on the validation sets. Therefore, we repeated all the experiments five times and reported the average performance.

6.5.2 Metrics

Effectiveness was measured by five standard evaluation metrics: mean reciprocal rank (MRR), precision of the top 1 retrieved app (P@1), normal discounted cumulative gain for the top 1, 3, and 5 retrieved apps (nDCG@1, nDCG@3,

nDCG@5). We determined the statistically significant differences using the two-tailed paired t-test with Bonferroni correction at a 95% confidence interval ($p < 0.05$). In the ranked list of apps associated to every query, we assigned the score of 2 to the *first* relevant app and 1 to the rest of relevant apps, to differentiate between a model that is able to rank the first relevant app higher and a model that is not.

The choice of evaluation metrics was motivated by considering three different aspects of the task, inspired by data analysis. We chose MRR considering scenarios where a user is looking for relevant information only in one app, and so they would stop scanning the search results as soon as they find the first relevant document. We reported P@1 and nDCG@1 to measure the performance for scenarios that a user only checks the first result. Given that many search tasks need to be addressed using more than one app, it is crucial to evaluate a system with respect to more than one relevant app in the top- k results. nDCG@3 allowed us to evaluate our approach when a user scans the top 3 results. Since we found that most of the queries were assigned to one or two apps (see Section 6.3), nDCG@3 measures how well a system is able to place the two relevant apps among the top 3 results. We also used nDCG@5 to evaluate top 5 results on a single screen, given the size of a typical smartphone.

6.5.3 Compared Methods

We compared the performance of our model with the following methods:

- *StaticRanker*: For every query we ranked the apps in the order of their popularity in the training set as a static (query independent) model.
- *QueryLM*, *BM25*, *BM25-QE*: For every app we aggregated all the relevant queries from the training set to build a document representing the app. Then we used Terrier [141] to index the documents. QueryLM uses the language model retrieval model [150]. For BM25-QE, we adopted Bo1 [24] model for query expansion. We used the Terrier implementation of these methods.
- *k-NN*, *k-NN-AWE*: To find the nearest neighbors in k nearest neighbors (k -NN), we considered the cosine similarity between TF-IDF vectors of queries. Then, we took the labels (apps) of the nearest queries and produced the app ranking. As for k -NN-AWE, we computed the cosine similarity between the average word embedding (AWE) of the queries obtained from GloVe [147] with 300 dimensions.

- *LambdaMART*: For every query-app pair, we used the scores obtained by BM25, k-NN, and k-NN-AWE as features to train LambdaMART [193] implemented in RankLib.⁷ For every query, we considered all irrelevant apps as negative samples.

6.6 Results and Discussion

In the following, we evaluate the performance of NTAS1 and NTAS2 trained on both data splits. We further analyze how other baseline models perform comparing their performance on both splits.

6.6.1 Performance Comparison

Table 6.4 lists the performance of our proposed methods as well as the compared methods. As we can see, the performance of all methods drops when we use UniMobile-T data splits, except for StaticRanker. StaticRanker gives us an idea of how much the test set is biased towards more popular apps. For example, we see that StaticRanker performs better on UniMobile-T suggesting that it consists of more popular apps. As we compare the relative performance drop between the two data splits, we see that among other baselines, k-NN-AWE is more robust with the minimum relative drop (−8.4% on average). QueryLM, on the other hand, is the least robust model with the maximum relative drop (−16% on average). This indicates that k-NN-AWE is able to capture similar queries for unseen tasks using a pre-trained word embedding, whereas QueryLM relies heavily on the indexed queries.

Among the baselines tested on UniMobile-Q, we see that BM25 performs best in terms of all evaluation metrics. Given that UniMobile-Q contains queries belonging to the same tasks both in training and test sets, this shows that when more similar queries exist in the index, BM25 is able to rank the apps more effectively. However, on UniMobile-T, k-NN-AWE performs best in terms of all metrics. Given that UniMobile-T *does not* contain queries belonging to the same task in training and test sets, this suggests that leveraging a pre-trained word embedding helps k-NN capture query similarities more effectively when the queries are less similar, leading to a better generalization. This can also be seen when comparing the performance of k-NN and k-NN-AWE, given that k-NN-AWE consistently outperforms k-NN. Regarding LambdaMART, we see that even though it benefits from multiple features, it does not perform as well as k-NN-AWE and BM25

⁷<https://sourceforge.net/p/lemur/wiki/RankLib/>

Table 6.4. Performance comparison with baselines on UniMobile-Q and UniMobile-T.

Method	UniMobile-Q					UniMobile-T				
	MRR	P@1	nDCG@1	nDCG@3	nDCG@5	MRR	P@1	nDCG@1	nDCG@3	nDCG@5
StaticRanker	0.6485	0.5293	0.4031	0.4501	0.5144	0.6718	0.5507	0.4247	0.4853	0.5446
QueryLM	0.5867	0.3803	0.3068	0.4676	0.5508	0.5178	0.3272	0.2619	0.3716	0.4503
BM25	0.7523	0.6233	0.4915	0.6298	0.6859	0.6780	0.5244	0.4101	0.5392	0.5992
BM25-QE	0.6948	0.5177	0.4116	0.5909	0.6498	0.6256	0.4276	0.3312	0.5015	0.5704
k-NN	0.7373	0.6031	0.4794	0.6091	0.6633	0.6879	0.5414	0.4287	0.5413	0.6003
k-NN-AWE	0.7420	0.6081	0.4842	0.6156	0.6682	0.6984	0.5551	0.4407	0.5560	0.6117
LambdaMART	0.7313	0.6127	0.4864	0.6110	0.6426	0.6749	0.5469	0.4323	0.5419	0.5704
NTAS1-pointwise	0.7591*	0.6214	0.4897	0.6328	0.6934*	0.7047*	0.5582*	0.4493*	0.5506*	0.6258*
NTAS1-pairwise	0.7661*	0.6285*	0.5012*	0.6364*	0.7018*	0.7192*	0.5661*	0.4709*	0.5941*	0.6471
NTAS2	0.7638*	0.6271*	0.4996*	0.6351*	0.6976*	0.7144*	0.5723*	0.4608*	0.5689*	0.6334*

The superscript * denotes significant differences compared to all the baselines.

on UniMobile-Q. On the contrary, we see that it performs better on UniMobile-T showing that the AWE-based feature improves its generalization.

As we can see, NTAS1-pairwise and NTAS2 outperform all the methods, on both data splits, in terms of all evaluation metrics. All the improvements are statistically significant suggesting that using queries to learn the app representation helps our approach learn the similarities more effectively. Considering the relative difference on the two data splits, we observe that our proposed approaches also show a drop. Compared to other methods (except for StaticRanker), we observe that NTAS1-pairwise and NTAS2 consistently have a lower relative drop across UniMobile-Q and UniMobile-T, indicating that the trained app embedding is an effective way to represent mobile apps based on the queries that are assigned to them. Among our proposed methods, NTAS1-pairwise has the least relative drop (-7.4% on average), suggesting that a pairwise setting leads to a higher generalization.

6.6.2 Representation Analysis

We reduce the dimensionality of the learned app representations by projecting them to a two-dimensional space using t-Distributed Stochastic Neighbor Embedding (t-SNE) [134]. Figure 6.9 shows the proximity of the representation of different apps⁸ being grouped in some clusters. For instance, all social media apps are placed close to each other. Also, we see that location search and navigation apps are in another cluster. Interestingly, Gmail is close to File Manager, Contacts, and WhatsApp. People usually search for attachments or their contacts using Gmail, explaining their proximity. Google Search, on the other hand, belongs to no cluster. This could be due to the variety of queries people submit to Google Search, placing it somewhere in the center of all other apps. However, we cannot explain why YouTube is close to WhatsApp, or why Play Store is close to Amazon. Hence, Figure 6.9 shows that learning high-dimensional app representation using the queries submitted to them is effective, though perhaps not perfect.

6.6.3 Performance on Apps

Here, we compute the mean performance of the queries targeted to a specific app and plot the result for each app in Figure 6.10. For the sake of visualization, we only compare the performance of NTAS1-pairwise with three other methods in

⁸Given space limitations, we could not include all the apps in this figure.

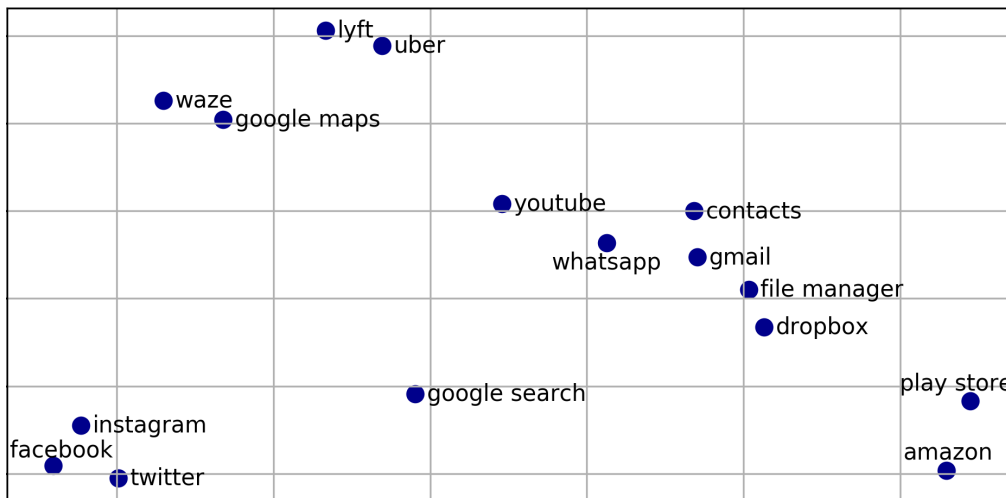
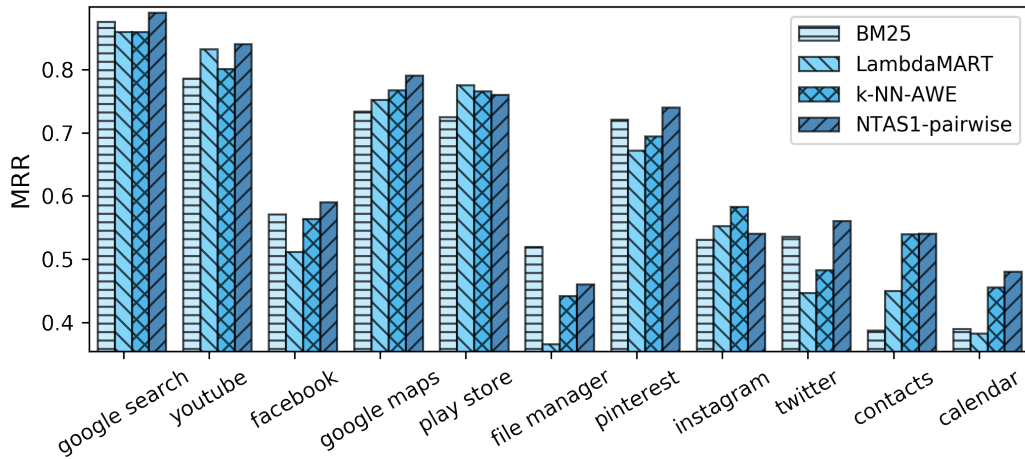


Figure 6.9. Proximity of different app representations learned by NTAS1-pairwise. This plot is produced by reducing the dimensionality (using the t-SNE algorithm) of the app representations to two for visualization.

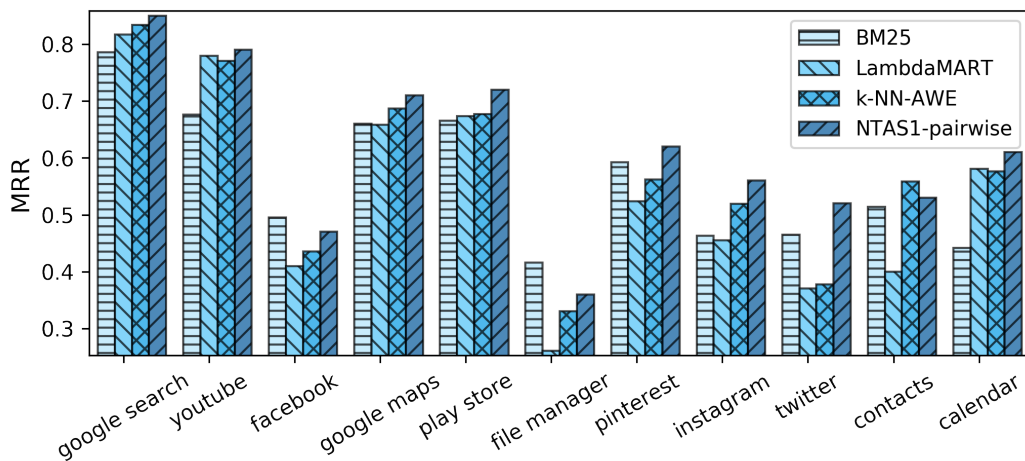
terms of MRR. We see that all models perform well in ranking less personal apps such as Google Search, YouTube, and Google Maps. Since none of the models incorporate users' personal data, this result is expected. This suggests users' personal data can be leveraged to rank apps such as File Manager, Contacts, and Calendar higher. Also, users' activities on their social media apps should be leveraged to provide a more effective personalized ranking. Moreover, we see that k-NN-AWE is more robust across the two data splits, compared to other baselines. In particular, it performs well in ranking Contacts suggesting that the proximity of contact names in the high-dimensional space of word embedding enables k-NN-AWE to outperform other models. Finally, it can be seen that NTAS1-pairwise is more robust across the two data splits, compared to other methods. Specifically, it outperforms all other methods for the majority of apps. However, NTAS1-pairwise performs worse than other methods for File Manager, on both data splits. This is mainly due to insufficient number of training data, given the diversity of the queries related to this app.

6.6.4 Performance on Tasks

We are interested in seeing how methods perform differently with respect to different search tasks. To do so, we averaged the performance (nDCG@3) of all queries belonging to the same task. Then, we grouped the tasks by the total



(a) UniMobile-Q



(b) UniMobile-T

Figure 6.10. Performance comparison with respect to certain apps on both data splits.

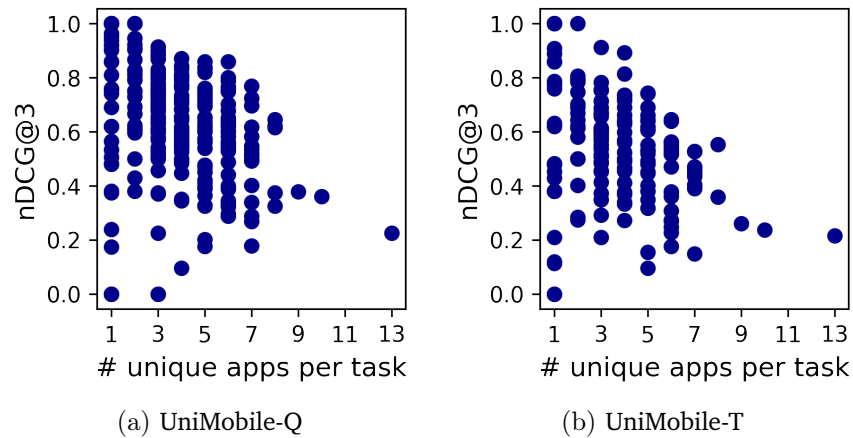


Figure 6.11. Negative correlation between the number of unique apps users selected for a task and performance.

number of unique apps selected by users and plotted their results in Figure 6.11. Our intuition was that if different users chose several apps for a single task, it can be a sign that the task is more challenging for the models. We can see in the figure that as the number of unique apps per task raises, the models perform worse. Although, the negative correlation is not very strong (Pearson’s $r = -0.3049$ and -0.3450 for UniMobile-Q and UniMobile-T, respectively), it is consistent with all models and evaluation metrics. This indicates that if a task can be done using multiple apps, their corresponding queries also become more difficult for a system. A multi-app task can be either very personal (i.e., every user chooses their own favorite app) or very general (i.e., it can be done using many apps). Therefore, one can explore incorporating users’ regular app usage patterns to perform a personalized target app selection.

6.7 Summary

In this chapter, we introduced and studied the task of target apps selection, which was motivated by the growing interest in conversational search systems where users speak their queries to a unified voice-based search system. To this aim, we presented the first analysis of mobile cross-app search queries and user behaviors in terms of the apps they chose to complete different search tasks. We found that a limited number of popular apps attract most of the search queries. We further observed notable differences between queries submitted to different apps. We

showed that query length and content differ among apps. We also showed that, 39% of search queries were done in Google Search, and it was the top choice of users in 35% of the tasks. Given that more than 71% of the defined tasks could be done with the current features of Google Search, this indicates that users prefer to search using a more specific app. We carried out the experiments and analyses on the dataset of cross-app mobile queries that we collected through crowdsourcing.

Since the mobile information environment is uncooperative and the data is heterogeneous, representing each app for the target apps selection task is challenging. We proposed two models that learn high-dimensional latent representations for the mobile apps in an end-to-end training setting. Our first model produces a score for a given query-app pair, while the second model produces a probability distribution over all the apps given a query. We compared the performance of our proposed method with state-of-the-art retrieval baselines splitting data following two different strategies. Our approach outperformed all baselines significantly.

After conducting a crowdsourced study of the unified mobile search task and finding out that such a task is needed in a mobile environment, we validate our findings in a more realistic situation where users report their everyday mobile queries through a bespoke app. The new setting not only provides a much more realistic experimental environment but also enables us to capture more-in-depth contextual information such as several sensor data as well as app usage statistics. In the next chapter, we will describe our effort for conducting a more realistic study as well as incorporating contextual information into a target apps selection model.

Chapter 7

Context-Aware Target Apps Selection

7.1 Introduction

As mobile devices provide rich contextual information about users [20], previous studies [4, 106, 211] have tried to incorporate query context in various domains. In particular, query context is often defined as the information from the previous queries and their corresponding clickthrough data [190, 194], or situational context such as location and time [39, 97, 211]. However, as user interactions on mobile devices are mostly with apps, exploring apps usage patterns reveals important information about the users contexts, information needs, and behavior. For instance, a user who starts spending time on travel-related apps, e.g., TripAdvisor, is likely to be planning a trip in the near future. Carrascal and Church [53] verified this claim by showing that people use certain categories of apps more intensely as they do mobile search.

However, as we described in the previous chapter our previous attempt to study unified mobile search through crowdsourcing failed to capture users' contexts while collecting data [19]. In addition, there are some other limitations. For example, we asked the workers to complete a set of given search tasks, which obviously were not their actual information needs, and thus the queries may differ from real search queries. In addition, the workers did not complete their work on mobile devices, which affects their behavior. Furthermore, the user behavior and queries could not be studied in a day-long or week-long period.

The aforementioned limitations have motivated us to conduct the first in situ study on target apps selection for unified mobile search. This enables us to obtain more clear insights into the task. In particular, we are interested in studying the users' behavior as they search for their real-life information needs using their own mobile devices. Moreover, we study the impact of contextual information

on the apps they use for search. To this aim, we developed a simple open source app, called uSearch, and used it to build an in situ collection of cross-app queries. Through an open call, we recruited 255 participants who installed uSearch and used it to report their queries as well as the target apps, right after they did a search on their smartphones. With participants' consents, uSearch also ran in the background collecting useful contextual data. We have released the code of uSearch to facilitate research on mobile information retrieval. In fact, uSearch is extendable and can be used for collecting data to study various search tasks on mobile devices. Over a period of 12 weeks, we collected thousands of queries which enables us to investigate various aspects of user behavior as they search for information in a cross-app search environment.

Using the collected data, we conduct an extensive data analysis, aiming to understand how users' behavior vary across different apps while they search for their information needs. The key findings of our analysis include the fact that users conduct the majority of their daily search tasks using specific apps, rather than Google. Among various available contextual information, we focus on the users' apps usage statistics as their *apps usage context*, and leave others for future work. This is motivated by the results of our analysis in which we show that users often search on the apps that they use more frequently. Based on the insights we got from our data analysis, we propose a context-aware neural target apps selection model, called CNTAS. In our model, we deal with the problem as a ranking task estimating a relevance score for a given context-query-app triplet. Our experiments demonstrate that our model significantly outperforms state-of-the-art retrieval models in this task. Also, we show that incorporating context improves nDCG@5 by an average of 20% on all models and improves the performance with respect to 57% of the users.

In summary, the main contributions of this chapter include:

- Designing and conducting an in situ mobile search study for collecting thousands of real-life cross-app queries. Both the app¹ and the collected data² are publicly available for research purposes.
- Presenting the first in situ analysis of cross-app queries and users' behavior as they search with different apps. More specifically, we study different attributes of cross-app mobile queries with respect to their target apps, sessions, and contexts.
- Proposing a context-aware neural model for target apps selection.

¹Available at <https://github.com/aliannejadi/uSearch>

²Available at <http://aliannejadi.com/istas.html>

- Evaluating the performance of state-of-the-art retrieval models for this task and comparing them against our proposed model.

Our analyses and experiments lead to new findings compared to previous studies, opening specific future directions in this research area.

7.2 Data Collection

In this section, we describe how we collected *ISTAS*, which is, to the best of our knowledge, the first in situ dataset on cross-app mobile search queries. We collected the data by recruiting 255 participants through an open call on the Web. The participants installed a simple Android app, called uSearch, for at least 24 hours on their smartphones. We asked them to use uSearch to report their real-life cross-app queries as well as the corresponding target apps.

We first describe the characteristics of uSearch. Then, we provide details on how we recruited participants as well as the details on how we instructed them to report queries through the app. Finally, we give details on how we checked the quality of the collected data.

7.2.1 uSearch

In order to facilitate the query report procedure, we developed uSearch, the Android app shown in Figure 7.1. We chose the Android platform because, in comparison with iOS, it imposes less restrictions in terms of sensor data collection and background app activity.

User interface. As shown in Figure 7.1, uSearch consists of three sections. The upper part lists all the apps that are installed on the phone, with the most used apps ranked higher. The participants were supposed to select the app in which they had done their real-life search (e.g., Facebook). In the second section, the participants were supposed to enter exactly the same query that they had entered in the target app (e.g., Facebook). Finally, the lower part of the app, provided them easy access to a unique ID of their device and an online survey on their demographics and backgrounds.

Collected data. Apart from the participants' input data, we also collected their interactions within uSearch (i.e., taps and scrolling). Moreover, a background service collected the phone's sensors data. We collected data from the following sensors: (i) GPS; (ii) accelerometer; (iii) gyroscope; (iv) ambient light; (v) WiFi; and (vi) cellular. Also, we collected other available phone data that can be used

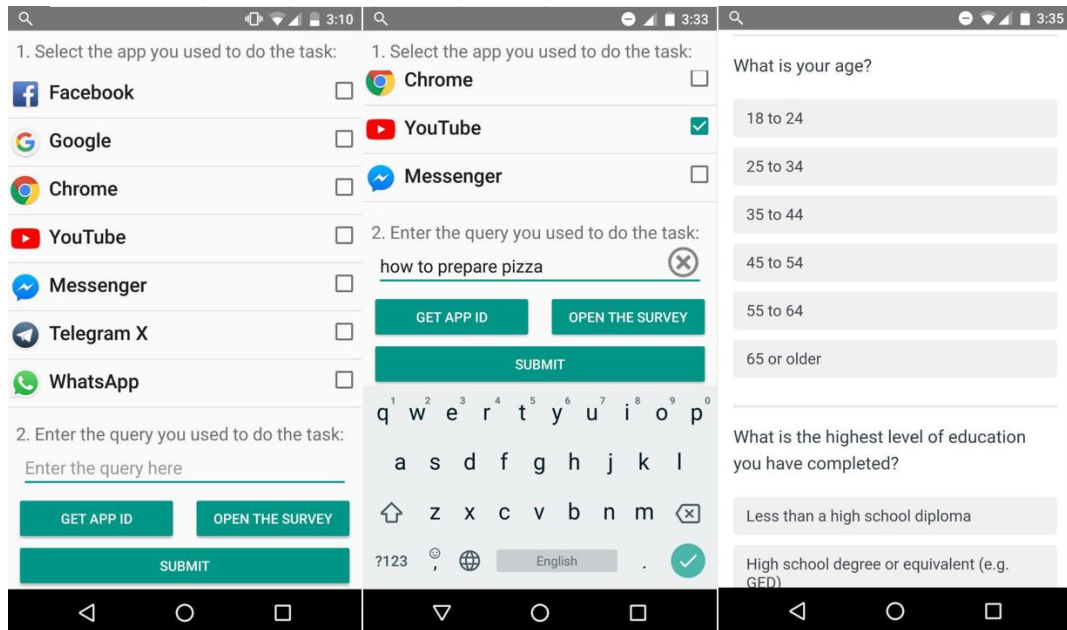


Figure 7.1. uSearch interface on LG Google Nexus 5 as well as the survey. Checkboxes are used to indicate the target app for a query.

to better understand a user's context. The additional collected data are as follows: (i) battery level; (ii) screen on/off events; (iii) apps usage statistics; and (iv) apps usage events. Note that apps usage statistics indicate how often each app has been used in the past 24 hours, whereas apps usage events provides more detailed app events.³ Apps usage events record user interactions in terms of: (i) launching a specific app; (ii) interacting with a launched app; (iii) closing a launched app; (iv) installing an app; and (v) uninstalling an app. The background service collected the data at a predefined time interval. The data was securely transferred to a cloud service.

7.2.2 Data Collection Procedure

We recruited participants through an online platform. In the announcement, we provided all the details about the intention of the study as well as the data we were collecting. First, we asked them to complete a survey inside uSearch. Moreover, we mentioned all the steps required to be done by the participants in order to report a query. In short, we asked them to open uSearch after every search they did using any installed app on their phones. Then, we asked them

³<https://developer.android.com/reference/android/app/usage/package-summary>

to report the app as well as the query they used to perform their search task. We encouraged the participants to report their search as soon as it occurs, as it is very crucial to capture their context at the right moment.

After running several pilot studies, over the period of 12 weeks we recruited 255 participants, asking them to let the app running on their smartphones for at least 24 hours and report at least 5 queries. Since some people may not submit 5 search queries during the period of 24 hours, we asked them to keep the app running on their phones after the first 24 hours until they reported 5 queries. Also, we encouraged them to continue reporting more than 5 queries for an additional reward. As incentive, we paid the participants \$0.2 per query. We recruited participants only from English-speaking countries.

7.2.3 Quality Check

During the course of data collection, we performed daily quality checks on the collected data. The checks were done manually with the help of some data visualization tools that we developed. As we were paying the participants a reward per query, we carefully studied the submitted queries as well as user interactions to prevent participants from reporting false queries. For each query, we checked the apps usage statistics and events for the same day. If a participant reported a query in a specific app (e.g., Facebook) but we could not find any recent usage events regarding that app, we assumed that the query was falsely reported. Moreover, if a participant reported more than 10 queries per day, we took some extra quality measures into account. Finally, we approved 6,877 queries out of 7,750 reported queries.

7.2.4 Privacy Concerns

Before asking for required app permissions, we made clear statement about our intentions on how we are going to use the participants' collected data as well as what was collected from their devices. We ensured them that their data was stored on secure cloud servers and that they could opt out at any point of the study. In that case we would remove all their data from the servers. While granting apps usage access was mandatory, granting location access was optional. We asked participants to allow uSearch access their locations only if they felt comfortable with that. Note that, through the background service, we did not collect any other data that could be used to identify the identity of participants.

7.2.5 Limitations

Like any other study, our study has some limitations. First, the study relies on self-reporting. This could result in specific biases in the collected data. For instance, participants may prefer to report shorter queries simply because it requires less typing. Also, in many cases, participants are likely to forget reporting queries or do not report all the queries that belong to the same session. Second, the reported queries are not actually submitted to a unified search system and users may formulate their queries differently in such setting. For example, in a unified system a query may be “videos of Joe Bonamassa” but in YouTube it may be “Joe Bonamassa.” Both limitations are due to lack of an existing unified mobile search app. Hence, developing such app is essential for constructing a more realistic collection.

7.3 Data Analysis

In this section, we describe the basic characteristics of ISTAS, and present a thorough analysis of target apps, queries, sessions, and context.

7.3.1 Basic Statistics

During the period of 86 days, with the help of 255 participants, we were able to collect 6,877 search queries and their target apps as well as sensor and usage data. The collected raw data is over 300 gigabytes. Here, we summarize the main characteristics of the participants based on the submitted surveys. 59% of the participants were female and 50% aged between 25 and 34. Participants were from all kinds of educational backgrounds ranging from high school diploma to PhD. In particular, 32% of them had a college degree, followed by 30% with a bachelor’s degree. Smartphone was the main device used for connecting to the Internet for 53% of the participants, followed by laptop (25%). Among the participants, 67% used their smartphones more often for personal reasons rather than work. Finally, half of the participants stated that they use their smartphones 4 hours a day or more. Table 7.1 lists basic characteristics of ISTAS. Moreover, Figure 7.2 shows the number of queries and active participants per day during the data collection period. Note that as shown in Figure 7.2, in the first half collection period, we were mostly developing the visualization tools and did not recruit many participants.

Table 7.1. Statistics of ISTAS.

# queries	6,877
# unique queries	6,262
# users	255
# unique apps	192
# search sessions	3,796
# days data collected	86
Mean queries per user	26.97 ± 50.21
Mean queries per session	1.81 ± 2.88
Mean queries per day	79.96 ± 101.27
Mean days of report per user	7.38 ± 15.95
Mean unique apps per user	5.14 ± 14.06
Mean query terms	3.00 ± 1.96
Mean query characters	17.53 ± 10.46

7.3.2 Apps

How apps are distributed. Figure 7.3 shows how queries are distributed with respect to the top 20 apps. We see that the top 20 apps account for 88% of the searches in ISTAS, showing that the app distribution follows a power-law. While Google and Chrome queries respectively attract 26% and 23% of the target apps, users conduct half (51%) of their search tasks using other apps. This finding is inline with what was shown in a previous work [19]. However, we observe a higher percentage of searches done using Google and Chrome apps. This can be due to two reasons: (i) ISTAS is collected in situ and on mobile devices, thus being more realistic; (ii) ISTAS queries reflect real-life information needs rather than a set of given search tasks, hence the information need topics are diverse. Moreover, we observe a notable variety of apps among the top 20 apps, such as Spotify and Contacts. We also see Google Play Store among the top target apps. This suggests that people use their smartphones to search for a wide variety of search tasks, most of which were done by apps other than Google or Chrome. It should also be noted that users seek the majority of their information needs on various apps, even though there exists no unified mobile search system on their smartphones, suggesting that they might even do a smaller portion of their searches using Google or Chrome, if a unified mobile search system was available on their smartphones.

How apps are selected. Here, we analyze the behavior of the participants as

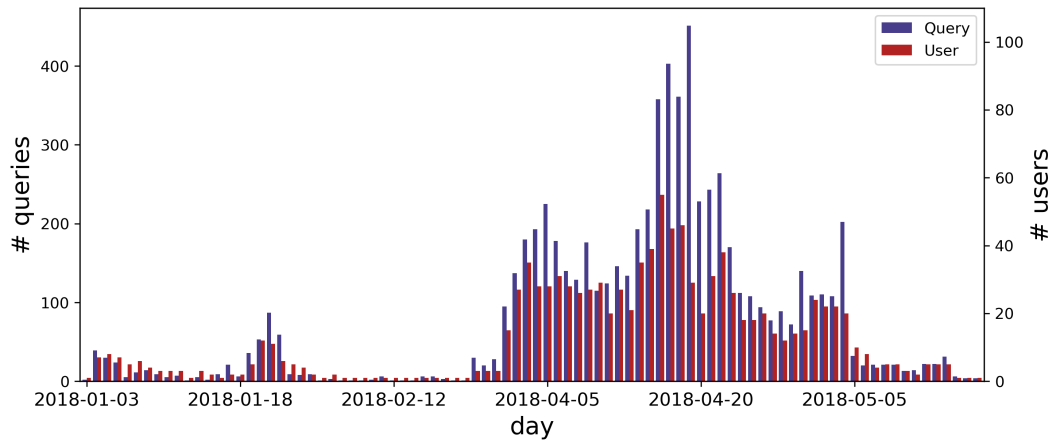


Figure 7.2. Number of queries and active participants per day, during the course of data collection (best viewed in color).

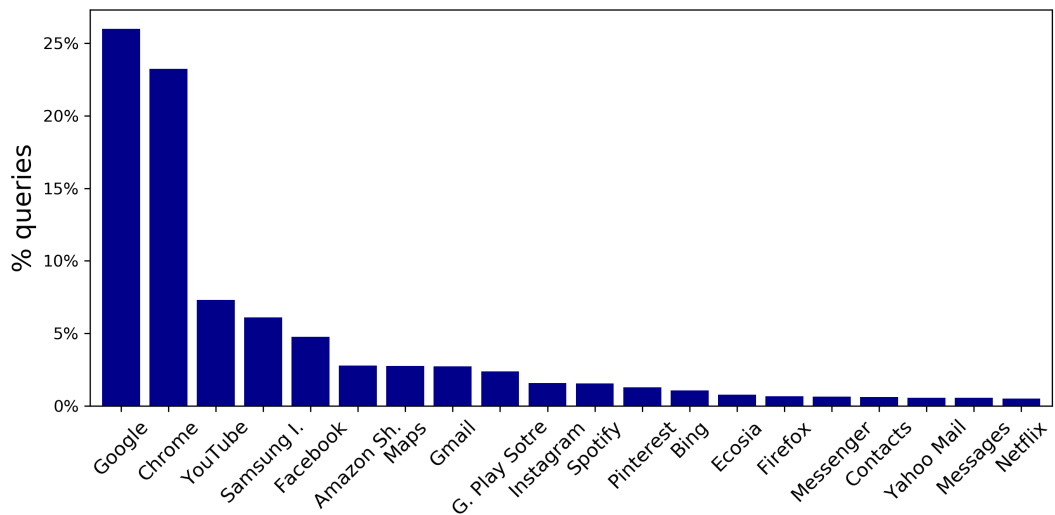


Figure 7.3. Number of queries per app for top 20 apps.

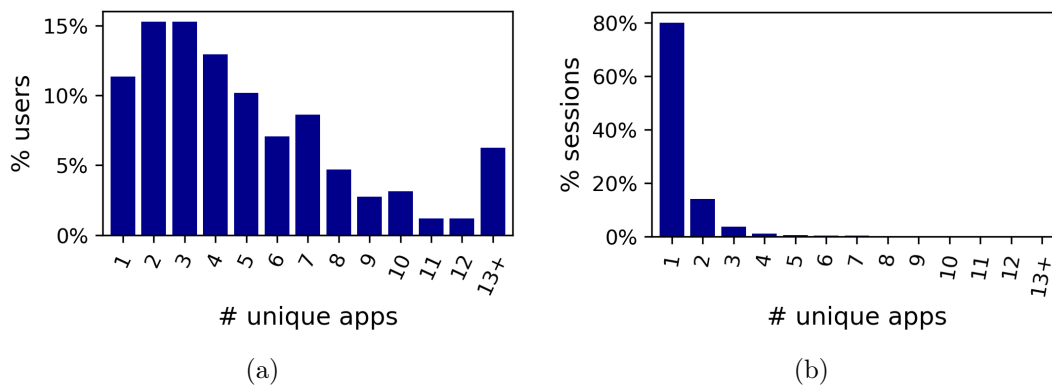


Figure 7.4. Distribution of unique apps per user and task.

they searched for real-life information needs, in terms of the apps they chose for performing the search. Figure 7.4a shows the distribution of unique apps per user. We can see how many users selected a certain number of unique apps, with an average of 5.14 unique apps per user. Again, this indicates that users seek information in a set of diverse apps. It is worth noting that in Figure 7.4a, we observe a totally different distribution compared to [19], where the average number of unique apps per user was much lower. We believe this difference is due to the fact that the participants in our work reported their real-life queries, as opposed to the crowdsourcing setup of [19].

On the other hand, Figure 7.4b plots the distribution of unique apps with respect to the sessions, that is how many unique apps were selected during a single search session. We see an entirely different distribution where the average number of unique apps per task is 1.36. This shows that while users seek information using multiple apps, they are less open to switching between apps in a single session. This can partly be due to the fact that switching between apps is not very convenient. However, this behavior requires more investigation that we leave for future work.

7.3.3 Queries

In order to understand the differences in user behavior while formulating their information needs using different apps, we conduct an analysis on the attributes of the queries with respect to their target apps. First, we start by studying the number of query terms in each app, for the top 9 apps in ISTAS.

How query length differs among apps. The upper part of Table 7.2 lists the distribution of the number of query terms in the whole dataset (denoted by *All*)

Table 7.2. Corss-app query attributes for 9 apps.

	All	Google	YouTube	Facebook	Amazon Sh.	Maps	Gmail	G. Play Store	Spotify	Contacts
# terms	Query term distribution									
1	22%	13%	11%	22%	12%	25%	57%	49%	29%	81%
2	28%	26%	29%	48%	45%	27%	30%	33%	35%	10%
3	20%	21%	24%	16%	25%	18%	9%	12%	24%	7%
4	12%	13%	18%	10%	10%	13%	3%	4%	7%	2%
> 4	17%	26%	17%	4%	10%	17%	1%	1%	6%	0%
Mean	3.00	3.49	3.19	2.34	2.74	3.07	1.61	1.75	2.31	1.31
τ	Query overlap									
> 0.25	56%	39%	41%	28%	27%	26%	27%	25%	8%	14%
> 0.50	19%	11%	15%	13%	7%	11%	12%	12%	4%	10%
> 0.75	13%	5%	8%	11%	5%	9%	12%	10%	2%	10%

The upper part lists the distribution of number of query terms as well as mean query terms per app.

The lower part lists the query overlap at different similarity thresholds (denoted by τ) per app.

All shows query distributions across all apps.

as well as each app. It also lists the average query terms per app. As we can see, the average query length is 3.00, which is slightly lower than previous studies on mobile query analysis [91, 105]. However, the average query length for apps that deal with general web search such as Google is higher (=3.49). This indicates that users submit shorter queries to other apps. For instance, we see that Contacts has the lowest average query length (=1.31). Also Gmail and Google Play Store have an average query length lower than 2. This difference shows a clear behavioral difference in formulating queries using different apps. Moreover, we can see that the distribution of the number of query terms varies among different apps; take Contacts as an example, whose single-term queries constitute 81% of its query distributions. This indicates that the structure of queries vary across the target apps. Studying the most frequent query unigrams of each app also confirms this finding. For example, Google's most popular unigrams are mostly stopwords (i.e., "to", "the", "of", "how"), whereas Facebook's most popular unigrams are not (i.e., "art", "eye", "wicked", "candy").

How query similarity differs across apps. The lower part of Table 7.2 lists the query similarity or query overlap using a simple function used in previous studies [19, 64]. We measure the query overlap at various degrees and use the similarity function $\text{sim}(q_1, q_2) = |q_1 \cap q_2| / |q_1 \cup q_2|$, simply measuring the overlap of query terms. We see that among all queries, 18% of them are similar to no other queries. We see a different level of query overlap in queries belonging to different apps. The highest overlap is among queries from Web search apps such as Chrome and Google. Lower query similarity is observed for personal apps such as Facebook and more focused apps such as Amazon Shopping. Note that the query overlap is higher when all app queries are taken into account (All), as opposed to individual apps. This shows that users submit more similar queries as they switch between apps, suggesting that switching between apps is part of the information seeking or query reformulation procedure on mobile devices.

7.3.4 Sessions

A session is a "series of queries by a single user made within a small range of time" [175]. Similar to previous work [53, 105, 175], we consider a 5-minute range of inactivity to close a session. ISTAS consists of 3,796 sessions, with 1.81 average queries per session. The majority of sessions have only one query (=66%). Similarly, as shown in Figure 7.4b, participants use only one app in the majority of sessions (=80%). We also studied how similar queries were distributed among single-app sessions as compared to multiple-app sessions. We

found that queries are more similar to each other in multiple-app sessions. More specifically, query overlap at the threshold of > 0.25 is 49% and 56% in single-app and multiple-app sessions, respectively. This suggests that users tend to switch between apps to search for the same information need as they reformulate their queries.

7.3.5 Context

Temporal behavior. We analyze the behavior of users as they search with respect to day-of-week and time-of-day. The distribution across day-of-week among the participants who reported their queries for more than 6 days slightly peaks on Fridays. Moreover, Figure 7.5 shows the distribution of queries and unique target apps across time-of-day for all participants. As we can see, more queries are submitted in the evenings, however we do not see a notable difference in the number of unique target apps.

Apps usage context. We define a user’s *apps usage context* at a given time t as the apps usage statistics of that specific user during the 24 hours before t . Apps usage statistics contain details about the amount of time users spent on every app installed on their smartphones. This gives valuable information on users’ personal app preferences as well as their contexts. For example, a user who has interacted with travel guide apps in the past 24 hours is probably planning a trip in the near future. Therefore, we analyze how users’ apps usage context can potentially help a target app selection model. Figure 7.6 shows the histogram of target app rankings in the users’ apps usage contexts. We see that participants often looked for information in the apps that they use more frequently. For instance, 19% of searches were done on the most used app, followed by 10% on the second most used app. We also see that, in most cases, as the ranking increases, the percentage of target apps decreases, suggesting that incorporating users app usage context is critical for target apps selection.

7.4 Context-Aware Neural Target Apps Selection

In this section, we propose a context-aware neural model called CNTAS, which is an extension to our recent neural target apps selection model (i.e., NTAS1) [19]. Our model takes a query q , a candidate app a , and the corresponding query context c_q as input and produces a score indicating the likelihood of the candidate app a being selected by the user as the target app for the query q .

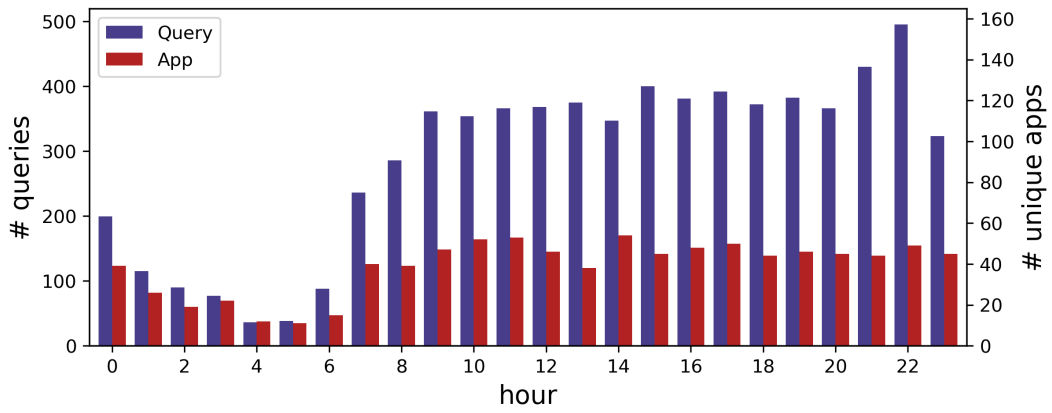


Figure 7.5. Time-of-the-day distribution of queries and unique apps (best viewed in color).

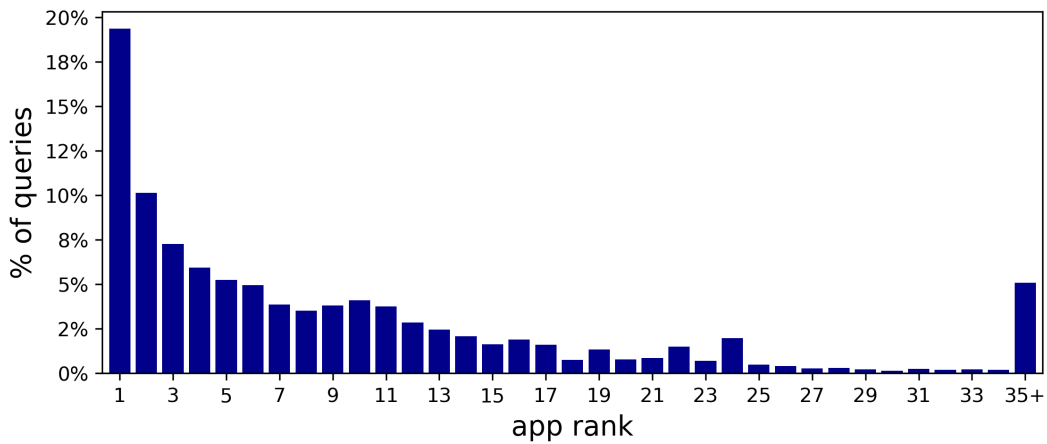


Figure 7.6. Apps usage context ranking distribution of relevant target apps. Lower values of x axis mean that the app has been used more often in the past 24 hours.

In the following, we first describe a *general* framework for context-aware target apps selection and further explain how it is implemented and how context is incorporated into the framework.

Formally, the CNTAS framework estimates the probability $p(S = 1|q, a, c_q; A)$, where S is a binary random variable indicating whether the app a should be selected ($S = 1$) or not ($S = 0$). A denotes the set of candidate apps. This set can be all the apps installed on the user’s mobile device, or the set of candidate apps that is obtained by another model in a cascade setting. The app selection probability in the CNTAS framework is estimated as follows:

$$p(S = 1|q, a, c_q; A) = \psi(\phi_Q(q), \phi_A(a), \phi_C(c_q)), \quad (7.1)$$

where ϕ_Q , ϕ_A , and ϕ_C respectively denote query representation, app representation, and context representation components. ψ is a target apps selection component that takes the mentioned representations and generates an app selection score. Any of these components can be implemented in different ways. In addition, c_q can contain various types of query context, including search time, search location, and the users apps usage.

We implement the component ϕ_Q with two major functions: an embedding function $\mathcal{E} : V \rightarrow \mathbb{R}^d$ that maps each vocabulary term to a d -dimensional embedding space, and a global term weighting function $\mathcal{W} : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The matrices \mathcal{E} and \mathcal{W} are the network parameters in our model and are learned to provide task-specific representations. The query representation component ϕ_Q represents a given query $q = \{w_1, w_2, \dots, w_{|q|}\}$ as follows:

$$\phi_Q(q) = \sum_{i=1}^{|q|} \widehat{\mathcal{W}}(w_i) \cdot \mathcal{E}(w_i),$$

which is the weighted element-wise summation over the terms’ embedding vectors. $\widehat{\mathcal{W}}$ is the normalized global weights computed using a softmax function as follows:

$$\widehat{\mathcal{W}}(w_i) = \frac{\exp(\mathcal{W}(w_i))}{\sum_{j=1}^{|q|} \exp(\mathcal{W}(w_j))}.$$

This is a simple yet effective approach for query representation based on the bag of words assumption, which has proven to be effective for target apps selection [19], ad-hoc retrieval [77], and query performance prediction [212].

To implement the app representation component ϕ_A , we learn a d -dimensional dense representation for each app. In more detail, this component consists of an

app representation matrix $\mathcal{A} \in \mathbb{R}^{N \times d}$ where N denotes the total number of apps. Therefore, $\phi_A(a)$ returns a row of the matrix \mathcal{A} that corresponds to the app a .

Various context definitions can be considered to implement the context representation component. General types of context, such as location and time, has been extensively explored in different tasks, such as web search [39], personal search [211], and mobile search [97]. In this document, we refer to the *apps usage time* as context, which is a special type of context for our task. As introduced earlier in Section 7.3.5, the apps usage context is the time that the user spent on each mobile app in the past 24 hours of the search time. To implement ϕ_C , we first compute a probabilistic distribution based on the apps usage context, as follows:

$$p(a'|c_q) = \frac{\text{time spent on app } a' \text{ in the past 24 hours}}{\sum_{a'' \in A} \text{time spent on app } a'' \text{ in the past 24 hours}},$$

where A is a set of candidate apps. ϕ_C is then computed as:

$$\phi_C(c_q) = \sum_{a' \in A} p(a'|c_q) \cdot \mathcal{A}_C[a'],$$

where $\mathcal{A}_C \in \mathbb{R}^{N \times d}$ denotes an app representation matrix which is different from \mathcal{A} used in the app representation component. This matrix is supposed to learn app representations suitable for representing the apps usage context. $\mathcal{A}_C[a']$ denotes the representation of app a' in the app representation matrix \mathcal{A}_C .

In summary, each of the representation learning components ϕ_Q , ϕ_A , and ϕ_C returns a d -dimensional vector. The app selection component is modeled as a fully-connected feed-forward network with two hidden layers and the output dimensionality of 1. We use rectified linear unit (ReLU) as the activation function in the hidden layers of the network. Sigmoid is used as the final activation function. To avoid overfitting, the dropout technique [179] is employed. For each query, the following vector is fed to this network:

$$(\phi_Q(q) \circ \phi_A(a)) \cdot |\phi_Q(q) - \phi_A(a)| \cdot (\phi_C(c_q) \circ \phi_A(a)) \cdot |\phi_C(c_q) - \phi_A(a)|,$$

where \circ denotes the Hadamard product, i.e., the element-wise multiplication, and \cdot here means concatenation. In fact, this component computes the similarity of the candidate app with the query content and context, and estimates the app selection score based on the combination of both.

We train our model using pointwise and pairwise settings. In a pointwise setting, we use mean squared error (MSE) as the loss function. MSE for a mini-batch b is defined as follows:

$$\mathcal{L}_{MSE}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} (y_i - \psi(\phi_Q(q_i), \phi_A(a_i), \phi_C(c_{q_i})))^2,$$

where q_i , c_{q_i} , a_i , and y_i denote the query, the query context, the candidate app, and the label in the i^{th} training instance of the mini-batch. For this training setting, we use a linear activation for the output layer.

CNTAS can also be trained in a pairwise fashion. Therefore, each training instance consists of a query, the query context, a target app, and a non-target app. To this end, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [123]. Hinge loss is a linear loss function that penalizes examples violating the margin constraint. For a mini-batch b , hinge loss is defined as below:

$$\mathcal{L}_{Hinge}(b) = \frac{1}{|b|} \sum_{i=1}^{|b|} \max\{0, 1 - \text{sign}(y_{i1} - y_{i2})(\hat{y}_{i1} - \hat{y}_{i2})\},$$

where $\hat{y}_{ij} = \psi(\phi_Q(q_i), \phi_A(a_{ij}), \phi_C(c_{q_i}))$.

7.5 Experimental Setup

7.5.1 Data

We evaluate the performance of our proposed models on the ISTAS dataset. We follow two different strategies to split the data: (i) In *ISTAS-R*, we randomly select 70% of the queries for training, 10% for validation, and 20% for testing set; (ii) In *ISTAS-T*, we sort the queries of each user chronologically and keep the first 70% of each user’s queries for training, the next 10% for validation, and the last 20% for testing set. *ISTAS-T* is used to evaluate the methods when information about users’ search history is available. To minimize random bias, for *ISTAS-R* we repeated the experiments 10 times and report the average performance. The hyper-parameters of all models were tuned based on the nDCG@3 value on the validation sets.

7.5.2 Metrics

Effectiveness is measured by four standard evaluation metrics that were also used in [19]: mean reciprocal rank (MRR), and normalized discounted cumulative gain for the top 1, 3, and 5 retrieved apps (nDCG@1, nDCG@3, nDCG@5). We determine the statistically significant differences using the two-tailed paired t-test with Bonferroni correction at a 95% confidence interval ($p < 0.05$).

7.5.3 Compared Methods

We compared the performance of our model with the following methods:

- *StaticRanker*: For every query we rank the apps in the order of their popularity in the training set as a static (query independent) model.
- *QueryLM*, *BM25*, *BM25-QE*: For every app we aggregate all the relevant queries from the training set to build a document representing the app. QueryLM is the query likelihood retrieval model [150]. For BM25-QE, we adopt Bo1 [24] for query expansion. We use the Terrier [141] implementation of these methods.
- *k-NN*, *k-NN-AWE*: To find the nearest neighbors in k nearest neighbors (k-NN), we consider the cosine similarity between the TF-IDF vectors of queries. Then, we take the labels (apps) of the nearest queries and produce the app ranking. As for k-NN-AWE [210], we compute the cosine similarity between the average word embedding (AWE) of the queries obtained from GloVe [147] with 300 dimensions.
- *ListNet*, *ListNet-CX*: For every query-app pair, we use the scores obtained by BM25-QE, k-NN, k-NN-AWE, and StaticRanker as features to train ListNet [50] implemented in RankLib⁴. For every query, we consider all irrelevant apps as negative samples. ListNet-CX also includes users' apps usage context as an additional feature.
- *NTAS*: A neural model approach that we designed for the target apps selection task in our previous work [19]. We use the NTAS1 model due to its superior performance compared to NTAS2.
- *Contextual baselines*: In order to carry out a fair comparison between CNTAS and other context-aware baselines, we apply a context filter to all non-contextual baselines. We create the context filter as follows: for every app α in the training samples of user u , we take the time that u has spent on α in the past 24 hours as its score. We then perform a linear interpolation with the scores of all the mentioned baselines. Note that all scores are normalized. All these models are denoted by a -CR suffix.

⁴<https://sourceforge.net/p/lemur/wiki/RankLib/>

7.6 Results and Discussion

In the following, we evaluate the performance of CNTAS trained on both data splits and study the impact of context on the performance. We further analyze how the models perform on both data splits.

7.6.1 Performance Comparison

Table 7.3 lists the performance of our proposed methods as well as the compared methods. First, we compare the relative performance drop between the two data splits. We see that almost all non-contextual models perform worse on ISTAS-T compared to ISTAS-R, whereas almost all context-aware models perform better on ISTAS-T. Among the non-contextual methods, ListNet is the most robust model with the least performance drop and k-NN-AWE is the only method that performs better on ISTAS-T (apart from StaticRanker). On the other hand, QueryLM exhibits the most performance drop (-27% on average), as opposed to Contextual-k-NN-AWE with the highest performance improvement on ISTAS-T ($+10\%$ on average). This indicates that k-NN-AWE is able to capture similar queries more effectively, whereas QueryLM relies heavily on the indexed queries. It should also be noted that StaticRanker performs better on ISTAS-T indicating that it is biased towards more popular apps.

Among the non-contextual baselines, we see that NTAS-pairwise performs best in terms of most evaluation metrics on both data splits, this is because it learns high dimensional app and query representations which help it to perform more effectively. We see that applying the contextual filter improves the performance of all models. These improvements are statistically significant in all cases, so are not shown in the table. Although this filter is very simple, it is still able to incorporate useful information about user context and behavior into the ranking. This also indicates the importance of apps usage context, as mentioned in Section 7.3.5. Among the context-aware baselines, we see that NTAS-pairwise-CR performs best in terms of MRR and nDCG@1, while k-NN-AWE-CR and ListNet-CR perform better in terms of other evaluation metrics. It should be noted that ListNet-CR performs better than ListNet-CX. This happens because ListNet-CX integrates the apps usage context as an additional feature, whereas ListNet-CR is the result of the combination of ListNet and the contextual filter.

We see that our proposed CNTAS outperforms all the baselines with respect to the majority of evaluation metrics. In particular CNTAS-pairwise exhibits the best performance. The achieved improvements in terms of MRR and nDCG@1 are statistically significant. The reason is that CNTAS is able to learn latent features

Table 7.3. Performance comparison with baselines on ISTAS-R and ISTAS-T.

Methods	ISTAS-R Dataset					ISTAS-T Dataset				
	MRR	nDCG@1	nDCG@3	nDCG@5	MRR	nDCG@1	nDCG@3	nDCG@5		
StaticRanker	0.4502	0.2597	0.4435	0.4891	0.4786	0.2884	0.4752	0.5173		
QueryLM	0.3556	0.2431	0.3534	0.3900	0.2706	0.1486	0.2713	0.3097		
BM25	0.4205	0.3134	0.4363	0.4564	0.3573	0.2447	0.3771	0.3948		
BM25-QE	0.4319	0.2857	0.4371	0.4727	0.3930	0.2411	0.4053	0.4364		
k-NN	0.4433	0.2761	0.4455	0.4811	0.4067	0.2294	0.3982	0.4655		
k-NN-AWE	0.4742	0.2937	0.4815	0.5211	0.4859	0.2950	0.4919	0.5392		
ListNet	0.5170	0.3330	0.5211	0.5623	0.5118	0.3219	0.5208	0.5572		
NTAS-pointwise	0.5221	0.3427	0.5231	0.5586	0.5162	0.3385	0.5162	0.5550		
NTAS-pairwise	0.5257	0.3468	0.5236	0.5618	0.5214	0.3427	0.5183	0.5580		
Context-Aware Methods										
StaticRanker-CR	0.4903	0.3015	0.4901	0.5268	0.5289	0.3576	0.5358	0.5573		
QueryLM-CR	0.4540	0.2773	0.4426	0.5013	0.4696	0.3023	0.4597	0.5145		
BM25-CR	0.5398	0.3653	0.5394	0.5871	0.5249	0.3496	0.5255	0.5723		
BM25-QE-CR	0.5215	0.3398	0.5223	0.5693	0.5230	0.3474	0.5260	0.5728		
k-NN-CR	0.4978	0.3114	0.4926	0.5431	0.5161	0.3481	0.4956	0.5555		
k-NN-AWE-CR	0.5144	0.3233	0.5142	0.5632	0.5577	0.3722	0.5612	0.6086		
ListNet-CR	0.5391	0.3544	0.5417	0.5845	0.5599	0.3780	0.5657	0.6037		
ListNet-CX	0.5349	0.3580	0.5343	0.5784	0.5019	0.3139	0.5153	0.5521		
NTAS-pointwise-CR	0.5532	0.3745	0.5580	0.5883	0.5627	0.3865	0.5663	0.5965		
NTAS-pairwise-CR	0.5576	0.3779	0.5568	0.5870	0.5683	0.3923	0.5661	0.6047		
CNTAS-pointwise	0.5614*	0.3833*	0.5592	0.5901	0.5702*	0.4146*	0.5655	0.5938		
CNTAS-pairwise	0.5637*	0.3861*	0.5586	0.5924*	0.5738*	0.4182*	0.5679*	0.6071		

The superscript * denotes significant differences compared to all the baselines.

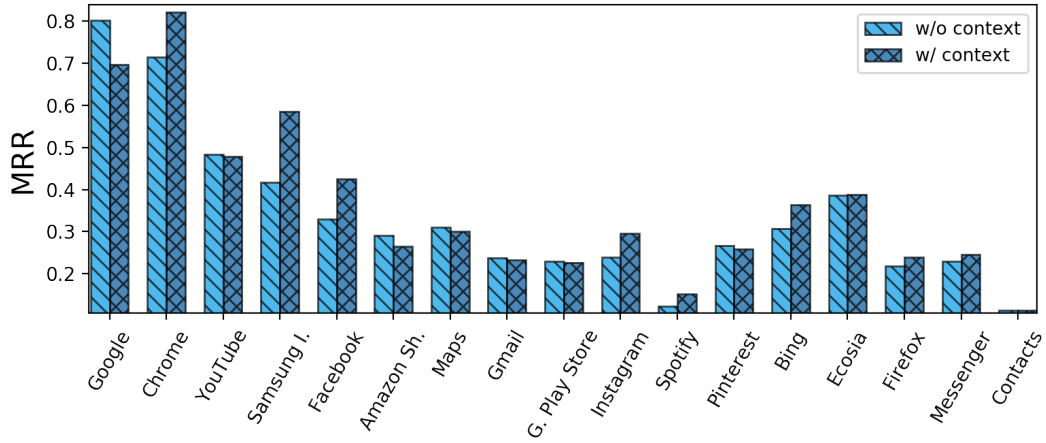


Figure 7.7. Performance comparison with respect to certain apps with and without context.

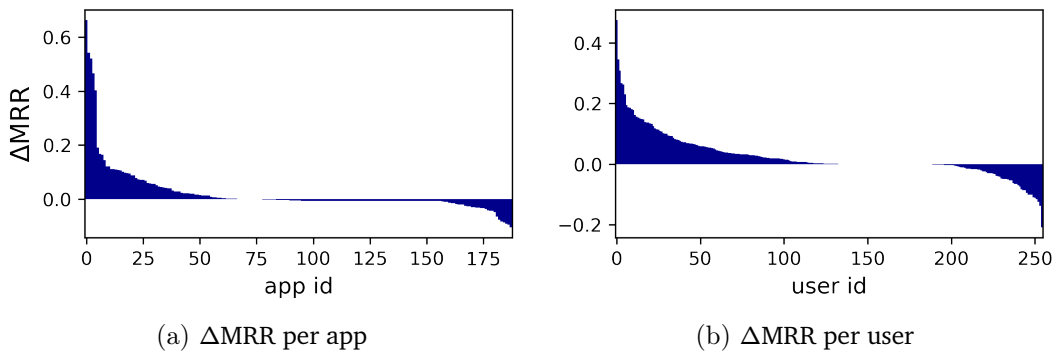


Figure 7.8. MRR differences on ISTAS-R with and without context per app and user.

Table 7.4. Performance analysis based on query length, dividing the test queries into three evenly-sized length buckets.

	Short queries	Med. queries	Long queries
	MRR	MRR	MRR
w/o context	0.5302	0.4924	0.4971
w/ context	0.5733	0.5190	0.4977

from the interaction of mobile usage data in the context. These interactions can reveal more information for better understanding of the user information needs.

7.6.2 Impact of Context on Performance Per App

In this experiment we demonstrate the effect of context on the performance with respect to various apps. Figure 7.7 shows the performance for queries that are labeled for specific target apps (as listed in the figure). We see that the context-aware model performs better while predicting social media apps such as Facebook and Instagram. However, we see that the performance for Google drops as it improves for Chrome. This happens because users do most of their browsing activities on Chrome, rather than Google; hence the usage statistics of Chrome helps the model to predict it more effectively. Moreover, we study the difference of MRR between the model with and without context for all apps. Our goal is to see how context improves the performance for every target app. We see in Figure 7.8a that the performance is improved for 39% of the apps. As shown in the figure, the improvements are much larger compared with the performance drops. Among the apps with the highest context improvements, we can mention Quora, Periscope, and Inbox. We saw that these apps were less popular among our participants and their representations were weaker than the other apps, that is why the contextual information shows the highest improvement for them.

7.6.3 Impact of Context on Performance Per User

Here we study the difference of MRR between the model with and without context for all users. Our goal is to see how many users are impacted positively by incorporating context in the target apps selection model. Figure 7.8b shows how performance differs per user when we apply context compared with when we do not. As we can see, users' apps usage context is able to improve the ef-

fectiveness of target apps selection for the majority of users. In particular, the performance for 57% of the users is improved by incorporating the apps usage context. In fact, we observed that users with the highest impact from context use less popular apps.

7.6.4 Impact of Context on Performance Per Query Length.

Following Zamani et al. [213], we create three buckets of test queries based on query length uniformly. Therefore, the buckets will have approximately equal number of queries. The first bucket, called Short queries, contains the shortest queries. The second one, called Med. queries, constitutes of medium-length queries. The last bucket, called Long queries, includes the longest queries of our test set. Table 7.4 lists the performance of the model with and without context in terms of MRR. As we see, the average MRR for all three buckets is improved as we apply context. However, we observe that as the queries become shorter, the improvement increases. The reason is that shorter queries tend to be more general or ambiguous, and thus query context can have higher impact on improving search for these queries.

7.7 Summary

In this chapter, we conducted the first in situ study on the task of target apps selection, which was motivated by the growing interest in intelligent assistants and conversational search systems where users interact with a universal voice-based search system. To this aim, we developed an Android app, called uSearch, and recruited 255 participants, asking them to report their real-life cross-app mobile queries via uSearch. We observed notable differences in length and structure among queries submitted to different apps. Furthermore, we found that while users choose to search using various apps, few apps attract most of the search queries. We found that even though Google and Chrome are the most popular apps, users do only 26% and 23% of their searches in these apps, respectively. The in situ data collection enabled us to collect valuable information about users' contexts. For instance, we found that the target app for 29% of the queries were among the top two most used apps of a particular user. Inspired by our data analysis, we proposed a model that learns high-dimensional latent representations for the apps usage context and predicts the target app for a query. The model was trained with an end-to-end setting. Our model produces a score for a given context-query-app triplet. We compared the performance of our proposed

method with state-of-the-art retrieval baselines splitting data following two different strategies. We observed that our approach outperforms all baselines, significantly.

This chapter brings the thesis to the end of its part on a unified mobile search where we introduced and analyzed distributed IR in a mobile environment. Next, we present our work on conversational search. Increasing interest and recent advances in intelligent assistants make research on conversational search and recommendation inevitable. Moreover, as we mentioned earlier, distributed search in a mobile environment is an essential component of a functional intelligent agent since users would communicate their information needs through a single channel. That being said, in the next chapter, we take the first step in studying and formalizing the task of *asking clarifying questions* in conversational search systems.

Part III

Conversational Search

Chapter 8

Conversational Search with Clarifying Questions

8.1 Introduction

While searching on the Web, users often fail to formulate their complex information needs in a single query. As a consequence, they may need to scan multiple result pages or reformulate their queries. Alternatively, systems can decide to proactively ask questions to clarify users' intent before returning the result list [44, 153]. In other words, a system can assess the level of confidence in the results and decide whether to return the results or ask users questions to *clarify* their information need. The questions can be aimed to clarify ambiguous, faceted or incomplete queries [185]. Asking clarifying questions is especially important in conversational search systems for two reasons: (i) conversation is the most convenient way for natural language interactions and for asking questions [111] and (ii) a conversational system can only return a limited number of results, thus being confident about the retrieval performance becomes even more important. One possible approach to improve the confidence is to ask clarifying questions. Figure 8.1 shows an example of such a conversation selected from our dataset. We see that both users, Alice and Robin, issue the same query, "dinosaur." Assuming that the system does not have access to any prior personal or contextual information, the conversation starts with the same clarifying question. The rest of the conversation, however, depends on the users' responses. In fact, the users' responses aid the system to get a better understanding of the underlying information need.

A possible workflow for an information system with clarifying questions is shown in Figure 8.2. As we can see, Alice initiates a conversation by submitting
















 <div style="border: 1px solid black; padding: 2px; display: inline-block;">dinosaur </div> <p>Information Need (Facet) I'm looking for the Discovery Channel's dinosaur site, which has pictures of dinosaurs and games.</p>	 <div style="border: 1px solid black; padding: 2px; display: inline-block;">dinosaur </div> <p>Information Need (Facet) I'm looking for a list of all (or many of) the different kinds of dinosaurs, with pictures.</p>
 <p>Are you looking for dinosaur books?</p>	 <p>Are you looking for dinosaur books?</p>
 <p>No, just the discovery channel website.</p>	 <p>Yes, if they contain pictures of all the different kinds of dinosaurs.</p>
 <p>Are you looking for meat-eating or plant-eating dinosaurs?</p>	 <p>Which dinosaurs are you interested in?</p>
 <p>I'm not sure. <input checked="" type="checkbox"/> No answer</p>	 <p>I'm interested in any and all dinosaurs.</p>
 <p>Would you like to see pictures or videos of dinosaurs?</p>	 <p>Do you want a list of dinosaurs names?</p>
 <p>I'd like to see pictures of dinosaurs on the discovery channels website.</p>	 <p>Yes, I would also like the list to include pictures of the dinosaurs.</p>

Figure 8.1. Example conversations with clarifying questions from our dataset, Qulac. As we see, both users, Alice and Robin, issue the same query (“dinosaur”), however, their actual information needs are completely different. With no prior knowledge, the system starts with the same clarifying question. Depending on the user’s answers, the system selects the next questions in order to clarify the user’s information need. The tag “No answer” shows that the asked question is not related to the information need. We asked the crowdworkers to answer each question given the original query and information need. In cases where the question required knowledge that was out of the scope of the information need, the workers would mark their answer with a “No answer” tag (see example).

her query to the system. The system then retrieves a list of documents and estimates its confidence on the result list (i.e., “Present Results?”). If the system is not sufficiently confident to present the results to the user, it then starts the process of asking clarifying questions. As the first step, it generates a list of candidate questions related to Alice’s query. Next, the system selects a question from the candidate question list and asks it from the user. Based on Alice’s answer, the system retrieves new documents and repeats the process.

In this chapter, we formulate the task of selecting and asking clarifying questions in open-domain information-seeking conversational systems. To this end, we propose an offline evaluation framework based on faceted and ambiguous queries and collect a novel dataset, called *Qulac*,¹ building on top of the TREC Web Track 2009-2012 collections. *Qulac* consists of over 10K question-answer pairs for 198 TREC topics consisting of 762 facets. Inspired from successful examples of crowdsourced collections [18, 23], we collected clarifying questions and their corresponding answers for every topic-facet pair via crowdsourcing. Our offline evaluation protocol enables further research on the topic of asking clarifying questions in a conversational search session, providing a benchmarking methodology to the community.

Our experiments on an oracle model show that asking only one good question leads to over 100% retrieval performance improvement. Moreover, the analysis of the oracle model provides important intuitions related to this task. For instance, we see that asking clarifying questions can improve the performance of shorter queries more. Also, clarifying questions exhibit a more significant effect on improving the performance of ambiguous queries, compared to faceted queries. We further propose a retrieval framework following the workflow of Figure 8.2, consisting of three main components: (i) question retrieval; (ii) question selection; and (iii) document retrieval. The question selection model is a simple yet effective neural model that takes into account both users’ queries and conversation context. We compare the question retrieval and selection models with competitive term-matching and LTR baselines, showing their ability to significantly outperform the baselines. Finally, to foster research in this area, we have made *Qulac* publicly available.²

¹*Qulac*, pronounced ku:lak, means *blizzard* and *wonderful* in Persian.

²Code and data are available at <https://github.com/aliannejadi/qulac>.

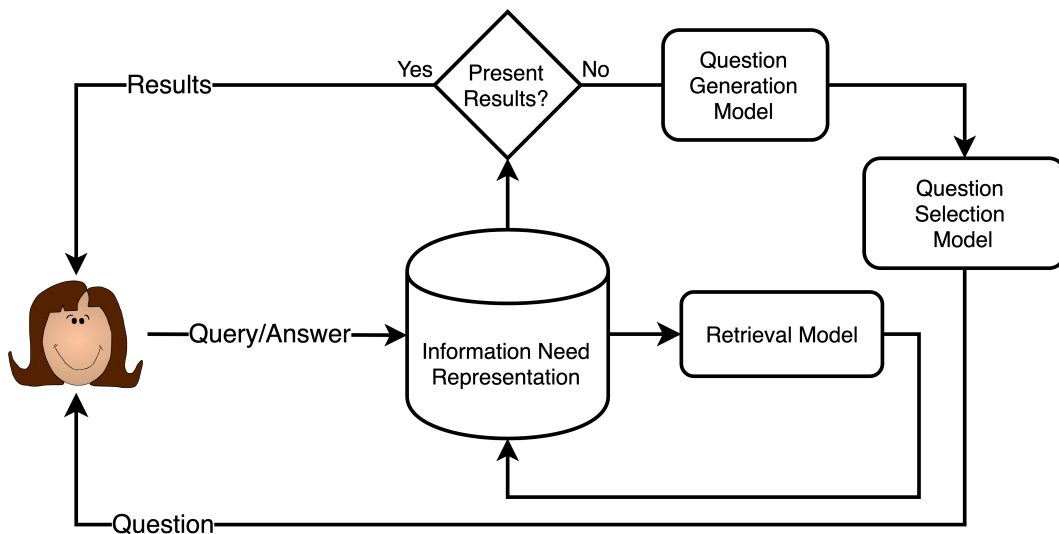


Figure 8.2. A workflow for asking clarifying questions in an open-domain conversational search system.

8.2 Problem Statement

A key advantage of a conversational search system is its ability to interact with the user in the form question and answer. In particular, a conversational search system can proactively pose questions to the users to understand their actual information needs more accurately and improve its confidence in the search results. We illustrate the workflow of a conversational search system, focusing on asking clarifying questions.

As depicted in Figure 8.2, once the user submits a query to the system, the *Information Need Representation* module generates and passes their information need to the *Retrieval Model*, which returns a ranked list of documents. The system should then measure its confidence in the retrieved documents (i.e., *Present Results?* in Figure 8.2). In cases where the system is not sufficiently confident about the quality of the result list, it passes the query and the context (including the results list) to the *Question Generation Model* to generate a set of clarifying questions, followed by the *Question Selection Model* whose aim is to select one of the generated questions to be presented to the user. Next, the user answers the question and the same procedure repeats until a stopping criterion is met. Note that when the user answers a question, the complete session information is considered for selecting the next question. In some cases, a system can decide to present some results, followed by asking a general question on possible further information. For example, assume a user submits the query “sigir 2019” and the

system responds “The deadline of SIGIR 2019 is Jan. 28. Would you like to know where it will be held?” As we can see, while the system is able to return an answer with high confidence, it can still ask further questions [196]. Here, we do not study this scenario; however, one can investigate it for exploratory search.

A facet-based offline evaluation protocol. The design of an offline evaluation protocol is challenging because conversation requires online interaction between a user and a system. Hence, an offline evaluation strategy requires human-generated answers to all possible questions that a system would ask, something that is impossible to achieve in an offline setting. To circumvent this problem, we substitute the Question Generation Model in Figure 8.2 with a large bank of questions, assuming that it consists of all possible questions in the collection. This assumption reduces the complexity of the evaluation significantly as human-generated answers to a limited set of questions can be collected offline, facilitating offline evaluation.

Here, we build our evaluation protocol on top of the TREC Web track’s data. TREC has released 200 search topics, each of which being either “ambiguous” or “faceted.”³ Clarke et al. [67] defined these categories as follows: “... Ambiguous queries are those that have multiple distinct interpretations. ... On the other hand, facets reflect underspecified queries, with different aspects covered by the subtopics...” The TREC collection is originally designed to evaluate search result diversification. In contrast, here we build various conversation scenarios based on topic facets.

Formally, let $\mathcal{T} = \{t_1, t_2, \dots, t_n\}$ be the set of topics (queries) that initiates a conversation. Moreover, we define $\mathcal{F} = \{\mathbf{f}_1, \mathbf{f}_2, \dots, \mathbf{f}_n\}$ as the set of facets with $\mathbf{f}_i = \{f_1^i, f_2^i, \dots, f_{m_i}^i\}$ defining different facets of t_i , where m_i denotes the number of facets for t_i . Further, let $\mathcal{Q} = \{\mathbf{q}_1, \mathbf{q}_2, \dots, \mathbf{q}_n\}$ be the set of clarifying questions belonging to every topic, where $\mathbf{q}_i = \{q_1^i, q_2^i, \dots, q_{z_i}^i\}$ consists of all clarifying questions that belong to t_i ; z_i is the number of clarifying questions for t_i . Here, our aim is to provide the users’ answers to all clarifying questions considering all topics and their corresponding facets. Therefore, let $\mathcal{A}(t, f, q) \rightarrow a$ define a function that returns answer a for a given topic t , facet f , and question q . Hence, to enable offline evaluation, \mathcal{A} requires to return an answer for all possible values of t , f , and q . In this chapter, \mathcal{T} and \mathcal{F} are borrowed from the TREC Web track 2009-2012 data. \mathcal{Q} is then collected via crowdsourcing and $\mathcal{A}(t, f, q)$ is also modeled by crowdsourcing (see Section 8.3). It is worth noting that we also borrow the relevance assessments of the TREC Web track,

³In this chapter, we use the term “facet” to refer to the subtopics of both faceted and ambiguous topics.

after breaking them down to the facet level. For instance, suppose the topic “dinosaur” has 10 relevant documents, 6 of which are labeled as relevant to the first facet, and 4 to the second facet. In Qulac, the topic “dinosaur” is broken into two topic-facet pairs together with their respective relevance judgments.

8.3 Data Collection

In this section, we explain how we collected Qulac (**Q**uestions for **l**ack of **c**larity), that is, to the best of our knowledge, the first dataset of clarifying questions in an IR setting. As we see in Figure 8.1, each topic is coupled with a facet. Therefore, the same question would receive a different answer based on the user’s actual information need. We follow a four-step strategy to build Qulac. In the first step we define the topics and their corresponding facets. In the second step, we collect a number of candidate clarifying questions (\mathcal{Q}) for each query through crowdsourcing. Then, in the third step, we assess the relevance of the questions to each facet and collect new questions for those facets that require more specific questions. Finally, in the last step, we collect the answers for every query-facet-question triplet, modeling \mathcal{A} . In the following subsections, we elaborate on every step of our data collection procedure.

8.3.1 Topics and Facets

As we discussed earlier, the problem of asking clarifying questions is particularly interesting in cases where a query can be interpreted in various ways. An example is shown in Figure 8.1 where two different users issue the same query for different intents. Therefore, any data collection should contain an initial query and description of its facet, describing the user’s information need. In other words, we define a target facet for each query. Faceted and ambiguous queries make an ideal case to study the effect of clarifying questions in a conversational search system for the following reasons: (i) the user information need is not clear from the query; (ii) multiple facets of the same query could satisfy the user’s information need; (iii) asking clarifying questions related to any of the facets provide a high information gain. Therefore, we choose the TREC Web track’s topics⁴ [69] as the basis for Qulac. In other words, we take the topics of TREC Web track 09-12 as initial user queries. We then break each topic down into its facets and assume that each facet describes the information need of a different user (i.e., it is a topic). Figures 8.3 and 8.4 illustrate how we have divided every topic into

⁴<https://trec.nist.gov/data/webmain.html>

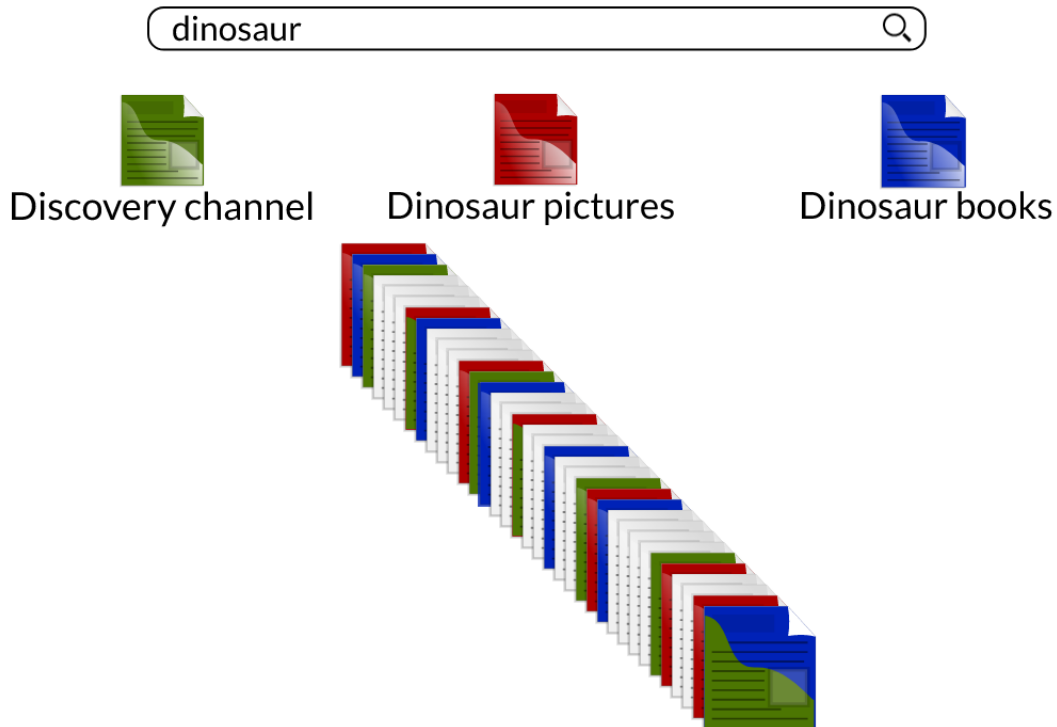


Figure 8.3. An example of three facets with their corresponding relevant documents for the topic “dinosaur” (best viewed in color).

its facets. As we can see in Figure 8.3, every TREC topic is assessed at a facet level. Therefore, all three colors (green, red, and blue) denote relevant documents to the topic “dinosaur,” each of which concerned with a different facet. Figure 8.4 shows how we have broken a topic into its facets and assumed that a user’s information need is one of them. Here, we see that Alice’s information need is concerned with the “Discovery channel,” hence only green documents that are labeled for this facet, are considered as relevant documents. We have broken down the topic to all its facets in a similar way as we can see for Robin and Ross.

As we see in Table 8.1, the average facet per topic is 3.85 ± 1.05 . Therefore, the initial 198 TREC topics⁵ leads to 762 topic-facet pairs in Qulac. Consequently, for each topic-facet pair, we take the relevance judgements associated with the respective facet.

⁵ The official TREC relevance judgements cover 198 of the topics.

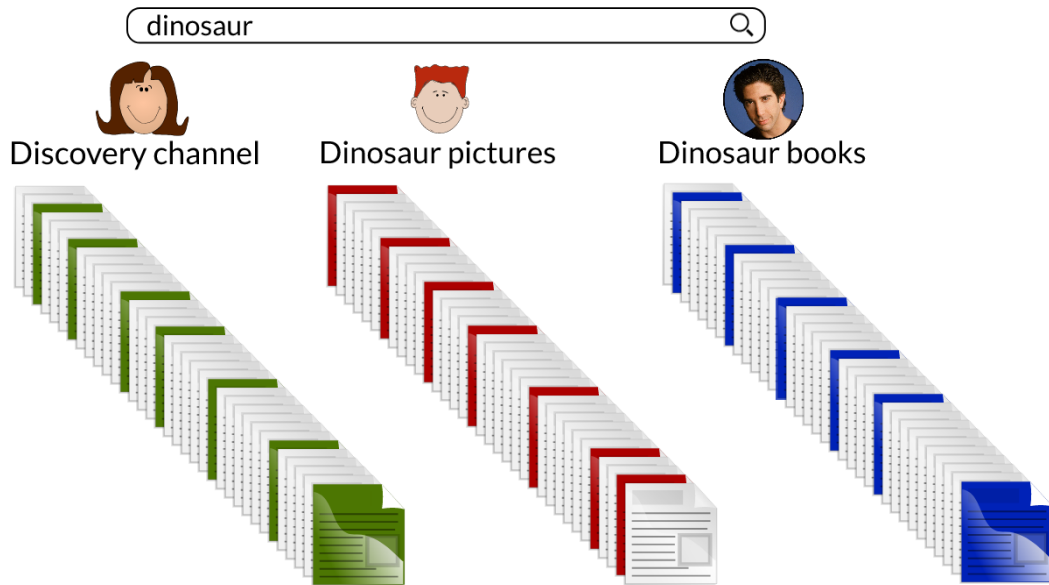


Figure 8.4. An example of three users who have issued the same query “dinosaur,” but with different information needs. As we see, the faceted relevance assessments are broken into three different sets creating three new topics (best viewed in color).

8.3.2 Clarifying Questions

It is crucial to collect a set of reasonable questions that address multiple facets of every topic⁶ while containing sufficient negative samples. This enables us to study the effect of retrieval models under the assumption of having a functional question generation model. Therefore, we asked human annotators to generate questions for a given query based on the results they observed on a commercial search engine as well as query auto-complete suggestions.

To collect clarifying questions, we designed a Human Intelligence Task (HIT) on Amazon Mechanical Turk.⁷ We asked workers to imagine themselves acting as a conversational agent such as Microsoft Cortana where an imaginary user had asked them about a topic. Then, we described the concept of facet to them, supporting it with multiple examples. Finally, we asked them to follow the steps below to figure out the facets of each query and generate questions accordingly:

1. Enter the same query in a search engine of their choice and scan the results in the first three pages. Reading the title of the results as well as scanning the

⁶Candidate clarifying questions should also address out-of-collection facets.

⁷<http://www.mturk.com>

snippets would give them an idea of different facets of the query on the Web.

2. For some difficult queries such as “toilet,” scanning the results would not help in identifying the facets. Therefore, inspired by [38], we asked the workers to type the query in the search box of the search engine, and press the space key after typing the query. Most commercial search engines provide a list of query auto-complete suggestions. Interestingly, in most cases the suggested queries reflect various aspects of the same query.
3. Finally, we asked them to generate six questions related to the query, aiming to address the facets that they had figured out.

The screenshots of the instructions that we provided to the workers can be found in Figure 8.5. We assigned two workers to each HIT, resulting in 12 questions per topic in the first round. In order to preserve the language diversity of the questions, we limited each worker to a maximum of two HITs. HITs were only available to workers residing in the U.S. with an approval rate of over 97%. After collecting the clarifying questions, in the next step, we explain how we verified them for quality assurance.

8.3.3 Question Verification and Addition

In this step, we aim to address two main concerns: (i) how good are the collected clarifying questions? (ii) are all facets addressed by at least one clarifying question? Given the high complexity of this step, we appointed two expert annotators for this task. We instructed the annotators to read all the collected questions of each topic, marking invalid and duplicate questions. Moreover, we asked them to match a question to a facet if the question was *relevant* to the facet. A question was considered relevant to a facet if its answer would address the facet. Finally, in order to make sure that all facets were covered by at least one question, we asked the annotators to generate an additional question for the facets that needed more specific questions. The outcome of this step is a set of verified clarifying questions, addressing all the facets in the collection.

8.3.4 Answers

After collecting and verifying the questions, we designed another HIT in which we collected answers to the questions for every facet. As we have illustrated in Figure 8.6, the HIT started with detailed instructions of the task, followed by

Instructions and examples (Click to collapse)

Task Description

Welcome and thanks for your interest. In this HIT, we would like to find out about different ways a search engine can ask questions from users to clarify their intention of a search. Therefore, here we show you a query which a user has submitted to a search engine (e.g., Google) and expect you to submit 6 clarifying questions. We highly recommend you to submit the same query to a search engine (Google, Bing, etc.), quickly scan the results and see what questions can help you distinguish between results.

Below you can see an example of clarifying question. The user has asked for "history of america", and the system asked the **clarifying question**: "Which historical period are you interested in?"

So, in this HIT, you should play the role of system and ask **clarifying questions** for the given queries.

 **history of america** ← query

 **Which historical period are you interested in?** ← clarifying question

 **I'd like to know more about the civil wars.** ← answer

 **Sure, here are some useful information on civil wars...** ← search result

IMPORTANT HINT

If you find it difficult to figure out different facets of a query by scanning the results, you can use the following trick for a better understanding of different facets:

- Open Google and type the given query in the search input box.
- As soon as you enter the last letter of the query and hit the space key, you will see Google's suggestions for the next word in the query.
- Those words usually show how the same query can be completed and cover different facets of the same query.
- Take the following figure as an example:



- As you see, the suggestions made by Google can give you good idea on various facets of the query "history of america". For example, one might be interested in buying a **book** on this topic, while someone might be looking for a **documentary**.
- This observation, together with scanning the results can give you an overview of different facets which should be helpful for generating clarifying questions.

Figure 8.5. Screenshots of clarifying question generation HIT instructions.

Instructions and examples (Click to collapse)

Task Description:

Welcome and thanks for your interest. This HIT is part of a joint research project between Università della Svizzera italiana in Switzerland and University of Massachusetts Amherst.

In this HIT, we would like to collect possible answers to questions that a conversational search system can ask users. Therefore, we assume that queries are submitted to a system that is able to interact with users. Such systems are Amazon Echo, Apple Siri, and Google Assistant.

Each HIT comes with an initial query that a user has submitted to the system. Then, there is a list of questions that a system has asked the user in order to understand their main intention behind the submitted query. All you need to do is to read the initial query and the user's information need. Then you should put yourself in the user's shoes and understand what the user is searching for. Based on that you should try to answer the questions to direct the search system to its main objective and clarify the search.

What you need to do:

- You are given a search query that a user has submitted. Assume the user has submitted this query to an interactive search engine (e.g., Apple Siri or Google Assistant).
- Read the user's actual information need of the query. This will give you an idea of why the user has submitted the query to the system.
- Read the clarifying question that the system has asked the user.
- Write an answer in natural language (as if you are having a conversation with Siri or Google Assistant) to the question.
- Assume that this is your actual information need, your answer should enable the system to understand the information need better.
- If you cannot answer the question based on the given Information Need, use answers such as "I don't know" or "This is not related to my search" and mark the question as "I cannot answer the question based on the given information need."

****** BONUS ******

We plan to launch several batches of this HIT in the coming days. After launching the final batch, we will send **ten \$1 bonuses** to ten workers with most contributions (with highest number of approved HITs). Since each worker is allowed to complete a maximum of 100 tasks, we will pick the 10 top workers randomly from those who have reached the 100 limit.

IMPORTANT:

- Please answer the questions only based on the given **Information Need**. Try to disregard your own preference or context while answering the questions.
- Please enter a complete answer in the form of natural language.
- Your answer is a part of a conversation. So, try to be engaging in the conversation as naturally as possible.
- Each worker is allowed to complete a maximum of 100 tasks. You will get an error message as soon as you reach the limit.

Figure 8.6. Screenshot of answer generation HIT instructions.

several examples. The workers were provided with a topic and a facet description. Then we instructed them to assume that they had submitted the query with their actual information need being the given facet. Finally, they were required to write the answer to *one* clarifying question that was presented to them. To avoid the bias of other questions for the same facet, we included only one question in each HIT. If a question required information other than what workers were provided with, we instructed the workers to identify it with a “No answer” tag. Each worker was allowed to complete a maximum of 100 HITs to guarantee language diversity. Workers were based in the U.S. with an approval rate of 95% or greater.

Quality check. During the course of data collection, we performed regular quality checks on the collected answers. The checks were done manually on 10% of submissions per worker. In case we observed any invalid submissions among the sampled answers of one user, we then studied all the submissions of that user. Invalid submissions were then removed from the collection and the worker was banned from the future HITs. Finally, we assigned all invalid answers to other workers to complete. Notice that we employed basic behavioral check

Table 8.1. Statistics of Qulac.

# topics	198
# faceted topics	141
# ambiguous topics	57
# facets	762
Average facet per topic	3.85 ± 1.05
Median facet per topic	4
# informational facets	577
# navigational facets	185
# questions	2,639
# question-answer pairs	10,277
Average terms per question	9.49 ± 2.53
Average terms per answer	8.21 ± 4.42

techniques in the design of the HIT. For example, we disabled copy/paste features of text inputs and tracked workers' keystrokes. This enabled us to detect and reject low-quality submissions.

8.4 Conversational Retrieval Framework

In this section, we propose a conversational search system that is able to select and ask clarifying questions and rank documents based on the user's responses. The proposed system initially retrieves a set of questions for a given query from a large pool of questions, containing all the questions in the collection. At the second stage, our proposed model, called NeuQS, selects the best question to be posed to the user based on the query and the conversation context. This problem is particularly challenging because the conversational interactions are in natural language, highly dependent on the previous interactions between the user and the system (i.e., conversation context).

As mentioned earlier in Section 8.2, a user initiates the conversation by submitting a query. Then the system should decide whether to ask a clarifying question or present the results. At every stage of the conversation, the previous questions and answers exchanged between the user and the system are known to the model. Finally, the selected question and its corresponding answer should be incorporated in the document retrieval model to enhance the retrieval performance.

Formally, for a given topic t let $\mathbf{h} = \{(q_1, a_1), (q_2, a_2), \dots, (q_{|\mathbf{h}|}, a_{|\mathbf{h}|})\}$ be the history of clarifying questions and the corresponding answers exchanged between the user and the system (i.e., the context). Here, the ultimate goal is to predict q , that is the next question that the system should ask the user. Moreover, let a be the user’s answer to q . The answer a is unknown to the question selection model, however, the document retrieval model retrieves documents once the system receives the answer a . In the following, we describe the question retrieval model, followed by the question selection and the document retrieval models.

8.4.1 Question Retrieval Model

We now describe **BERT**⁸ Language Representation based **Q**uestion **R**etrieval model, called BERT-LeaQuR. We aim to maximize the recall of the retrieved questions, retrieving all relevant clarifying questions to a given query in the top k questions. Retrieving all relevant questions from a large pool of questions is challenging, because questions are short and context-dependent. In other words, many questions depend on the conversation context and the query. Also, since conversation is in the form of natural language, term-matching models cannot effectively retrieve short questions. For instance, some relevant clarifying questions for the query “dinosaur” are: “Are you looking for a specific web page?” “Would you like to see some pictures?”

Yang et al. [197] showed that neural models outperform term-matching models for question retrieval. Inspired by their work, we learn a high-dimensional language representation for the query and the questions. Formally, BERT-LeaQuR estimates the probability $p(R = 1|t, q)$, where R is a binary random variable indicating whether the question q should be retrieved ($R = 1$) or not ($R = 0$). t and q denote the query (topic) and the candidate clarifying question, respectively. The question relevance probability in the BERT-LeaQuR model is estimated as follows:

$$p(R = 1|t, q) = \psi(\phi_T(t), \phi_Q(q)), \quad (8.1)$$

where ϕ_T and ϕ_Q denote topic representation and question representation, respectively. ψ is the matching component that takes the aforementioned representations and produces a question retrieval score. There are various ways to implement any of these components.

We implement ϕ_T and ϕ_Q similarly using a function that maps a sequence of words to a d -dimensional representation ($V^s \rightarrow \mathbb{R}^d$). We use the BERT [79] model to learn these representation functions. BERT is a deep neural network

⁸BERT: Bidirectional Encoder Representations from Transformers

with 12 layers that uses an attention-based network called Transformers [184]. We initialize the BERT parameters with the model that is pre-trained for the language modeling task on Wikipedia and fine-tune the parameters on Qulac with 3 epochs. Notice that BERT has recently outperformed state-of-the-art models in a number of language understanding and retrieval tasks [79, 142]. We particularly use BERT in our model to incorporate the knowledge from the vast amount of unlabeled data while learning the representation of queries and questions. In addition, BERT shows promising results in modeling short texts.

The component ψ is modeled using a fully-connected feed-forward network with the output dimensionality of 2. Rectified linear unit (ReLU) is employed as the activation function in the hidden layers, and a softmax function is applied on the output layer to compute the probability of each label (i.e., relevant or non-relevant). To train BERT-LeaQuR, we use a cross-entropy loss function.

8.4.2 Question Selection Model

In this section, we introduce a **Neural Question Selection Model (NeuQS)** which selects questions with a focus on maximizing the precision at the top of the ranked list. The main challenge in the question selection task is to predict whether a question has diverged from the query and conversation context. In cases where a user has given a negative answer(s) to previous question(s), the model needs to diverge from the history. In contrast, in cases where the answer to the previous question(s) is positive, questions on the same topic that ask for more details are preferred. For example, as we saw in Figure 8.1, when Robin answers the first question positively (i.e., being interested in dinosaur books), the second question tries to narrow down the information to a specific type of dinosaur.

NeuQS incorporates multiple sources of information. In particular, it learns from the similarity of a query, a question and the context as well as retrieval and performance prediction signals. In particular, NeuQS outputs a relevance score for a given query t , question q , and conversation context h . Formally, NeuQS can be defined as follows:

$$score = \gamma(\phi_T(t), \phi_H(\mathbf{h}), \phi_Q(q), \eta(t, \mathbf{h}, q), \sigma(t, \mathbf{h}, q)), \quad (8.2)$$

where γ is a scoring function for a given query representation $\phi_T(t)$, context representation $\phi_H(\mathbf{h})$, question representation $\phi_Q(q)$, retrieval representation $\eta(t, \mathbf{h}, q)$, and query performance representation $\sigma(t, \mathbf{h}, q)$. Various strategies can be employed to model each of the components of NeuQS.

We model the components ϕ_T and ϕ_Q similarly to Section 8.4.1. Further, the

context representation component ϕ_H is implemented as follows:

$$\phi_H(\mathbf{h}) = \frac{1}{|\mathbf{h}|} \sum_i^{|\mathbf{h}|} \phi_{QA}(q_i, a_i), \quad (8.3)$$

where $\phi_{QA}(q, a)$ is an embedding function of a question q and answer a . Moreover, the retrieval representation $\eta(t, \mathbf{h}, q) \in \mathbb{R}^k$ is implemented by interpolating the retrieval score of the query, context and question (see Section 8.4.3) and the score of the top k retrieved documents is used. Finally, the query performance prediction (QPP) representation component $\sigma(t, \mathbf{h}, q) \in \mathbb{R}^k$ consists of the performance prediction score of the ranked documents at different ranking positions (for a maximum of k ranked documents). We employed the σ QPP model for this component [148]. We take the representations from the [CLS] layer of the pre-trained uncased BERT-Base model (i.e., 12-layer, 768-hidden, 12-heads, 110M parameters). To model the function γ we concatenate and feed $\phi_T(t)$, $\phi_H(\mathbf{h})$, $\phi_Q(q)$, $\eta(t, \mathbf{h}, q)$, and $\sigma(t, \mathbf{h}, q)$ into a fully-connected feed-forward network with two hidden layers. We use ReLU as the activation function in the hidden layers of the network. We use a pointwise learning setting using a cross-entropy loss function.

8.4.3 Document Retrieval Model

Here, we describe the model that we use to retrieve documents given a query, conversation context, and current clarifying question as well as user's answer. We use the KL-divergence retrieval model [118] based on the language modeling framework [150] with Dirichlet prior smoothing [214] where we linearly interpolate two likelihood models: one based on the original query, and one based on the questions and their respective answers.

For every term w of the original query t , conversation context \mathbf{h} , the current question q , and answer a , the interpolated query probability is computed as follows:

$$p(w|t, \mathbf{h}, q, a) = \alpha \times p(w|\theta_t) + (1 - \alpha) \times p(w|\theta_{\mathbf{h},q,a}), \quad (8.4)$$

where θ_t denotes the language model of the original query, and $\theta_{\mathbf{h},q,a}$ denotes the language model of all questions and answers that have been exchanged in the conversation. α determines the weight of the original query and is tuned on the development set.

Then, the score of document d is calculated as follows:

$$p(d|t, \mathbf{h}, q, a) = \sum_{w_k \in \tau} p(w_k|t, \mathbf{h}, q, a) \log(p(w_k|d)), \quad (8.5)$$

where τ is the set of all the terms present in the conversation. We use Dirichlet’s smoothing for terms that do not appear in d . We use the document retrieval model for two purposes: (i) ranking documents after the user answers a clarifying question; (ii) ranking documents of a candidate question as part of the NeuQS (see Section 8.4.2). Hence, the model does not see the answer in the latter case.

8.5 Experimental Setup

8.5.1 Data

We evaluate BERT-LeaQuR and NeuQS on Qulac, following a 5-fold cross-validation. We follow two strategies to split the data:

- **Qulac-T:** we split the train/validation/test sets based on topics. In this case, the model has not seen the test topics in the training data;
- **Qulac-F:** here we split the data based on their facets. Thus, the same test topic might appear in the training set, but with a different facet.

In order to study the effect of multi-turn conversations with clarifying questions, we expand Qulac to include multiple artificially generated conversation turns. To do so, for each instance, we consider all possible combinations of questions to be asked as the context of conversation. Take t_1 as an example where we select a new question after asking the user two questions. Assuming that t_1 has four questions, all possible combinations of questions in the conversation context would be: $(q_1, q_2), (q_1, q_3), (q_1, q_4), (q_2, q_3), (q_2, q_4), (q_3, q_4)$. Notice that the set of candidate clarifying questions for each multi-turn example would be the ones that have not appeared in the context. The number of instances grows significantly as we enlarge the length of the conversation, leading to a total of 907,366 instances in the collection. At each turn of the conversation, we select the question from all candidate questions of the same topic and facet, having the same conversation history. In other words, they share the same context. Since the total number of unique conversational contexts is 75,200, a model should select questions for 75,200 contexts from all 907,366 candidate questions.

8.5.2 Metrics

Question retrieval evaluation metrics. We consider four metrics to evaluate the effectiveness of question retrieval models: mean average precision (MAP)

and recall for the top 10, 20, and 30 retrieved questions (Recall@10, Recall@20, Recall@30). Our choice of measures is motivated by the importance of achieving high recall for this task.

Question selection evaluation metrics. Effectiveness is measured considering the performance of retrieval after adding the selected question to the retrieval model as well as the user answer. To this end, at each stage of the conversation, we pass the selected question and its corresponding answer to the document retrieval model (Section 8.4.3). The model then re-ranks the documents based on the given query, question(s), and answer(s). Finally, the effectiveness of a question is evaluated based on the effect it has on the document retrieval performance. Five standard evaluation metrics are considered: mean reciprocal rank (MRR), precision of the top 1 retrieved document (P@1), and normalized discounted cumulative gain for the top 1, 5, and 20 retrieved documents (nDCG@1, nDCG@5, nDCG@20). We use the relevance assessments as they were released by TREC. However, we modify them in such a way to evaluate the performance with respect to every facet. For instance, if one topic consists of 4 facets it is then broken into 4 different topics each inheriting its own relevance judgements from the TREC assessments.

The choice of evaluation metrics is motivated by considering three different aspects of the task. We choose MRR to evaluate the effect of asking clarifying questions on ranking the first relevant document. We report P@1 and nDCG@1 to measure the performance for scenarios where the system is able to return only one result. This is often the case with voice-only conversational systems. Moreover, we report nDCG@5 and nDCG@20 as conventional ranking metrics to measure the impact of asking clarifying questions in a traditional Web search setting. Notice that nDCG@20 is the preferred evaluation metric for the ClueWeb collection due to the shallow pooling performed for relevance assessments [68, 133].

We determine statistically significant differences using the two-tailed paired t-test with Bonferroni correction at a 99.9% confidence interval ($p < 0.001$).

Compared Methods We compare the performance of our question retrieval and selection models with the following methods:

- Question retrieval:
 - *BM25*, *RM3*, *QL*: we index all the questions using Galago.⁹ Then, for a given query we retrieve the documents using BM25 [166], RM3 [120], and QL [150] models.

⁹<https://sourceforge.net/p/lemur/galago/>

- *LambdaMART, RankNet*: for every query-question pair, we use the scores obtained by BM25, RM3, and QL as features to train LambdaMART [193] and RankNet [46] implemented in RankLib.¹⁰ For every query, we consider all irrelevant questions as negative samples.
- Question selection:
 - *OriginalQuery* reports the performance of the document retrieval model only with the original query (Eq. (8.4) with $\alpha = 1$).
 - σ -*QPP*: we use a simple yet effective query performance predictor, σ [148] as an estimation of a question’s quality. We calculate the σ predictor of the document retrieval model with the following input: original query, the context, and candidate questions. We then select the question with the highest σ value.
 - *LambdaMART, RankNet*: we consider the task of question selection as a ranking problem where a list of candidate questions should be ranked and the one with the highest rank is chosen. Therefore, we use LambdaMART [193] and RankNet [46] as two LTR baselines. The list of features are: (i) a flag determining if a question is open or not; (ii) a flag indicating if the answer to the last question in the context is yes or no; (iii) σ [148] performance predictor of the current question; (iv) the Kendall’s τ correlation of the ranked list at 10 and 50 of the original query and the current question; (v) the Kendall’s τ correlation of the ranked list at 20 and 50 of the current question and previous question-answer pairs in the context; (vi) Similarity of the current question and the query based on their BERT representations; (vii) Similarity of the current question and previous question-answer pairs in the context based on their BERT representations.
 - *BestQuestion, WorstQuestion*: in addition to all the baselines, we also report the retrieval performance when the worst and the best question is selected for an instance. BestQuestion (or WorstQuestion) selects the candidate question for which the MRR value of the retrieval model is the maximum (or minimum). Note that the retrieval scores are calculated knowing the selected question and its answer (i.e., oracle model). Our goal is to show the upper and lower bounds.

¹⁰<https://sourceforge.net/p/lemur/wiki/RankLib/>

Table 8.2. Performance of question retrieval model.

Method	MAP	Recall@10	Recall@20	Recall@30
QL	0.6714	0.5917	0.6946	0.7076
BM25	0.6715	0.5938	0.6848	0.7076
RM3	0.6858	0.5970	0.7091	0.7244
LambdaMART	0.7218	0.6220	0.7234	0.7336
RankNet	0.7304	0.6233	0.7314	0.7500
BERT-LeaQuR	0.8349*	0.6775*	0.8310*	0.8630*

The superscript * denotes statistically significant differences compared to all the baselines ($p < 0.001$).

8.6 Results and Discussion

8.6.1 Question Retrieval

Table 8.2 shows the results of question retrieval for all the topics. As we see, BERT-LeaQuR is able to outperform all baselines. It is worth noting that the model’s performance gets better as the number of retrieved documents increases. This indicates that BERT-LeaQuR is able to capture the relevance of query and questions when they lack common terms. In fact, we see that all term-matching retrieval models such as BM25 are significantly outperformed in terms of all evaluation metrics.

8.6.2 Oracle Question Selection

Performance analysis. Here we study the performance of an oracle model, i.e. assuming that an oracle model is aware of the answers to the questions. The goal is to show to what extent clarifying questions can improve the performance of a retrieval system. As we see in the lower rows of Table 8.3 selecting the best questions (BestQuestion model) helps the model to achieve substantial improvement, even in the case that the retrieval model is very simple. This shows the high potential gain of asking good clarifying questions on the performance of a retrieval system. Particularly, we examine the relative improvement of the system after asking only one question and observe that BestQuestion achieves over 100% relative improvement in terms of different evaluation metrics (MRR: 0.2820 \rightarrow 0.5677, P@1: 0.1933 \rightarrow 0.4986, nDCG@1: 0.1460 \rightarrow 0.3988, nDCG@5: 0.1503 \rightarrow 0.2793, nDCG@20: 0.1520 \rightarrow 0.2265). It is worth men-

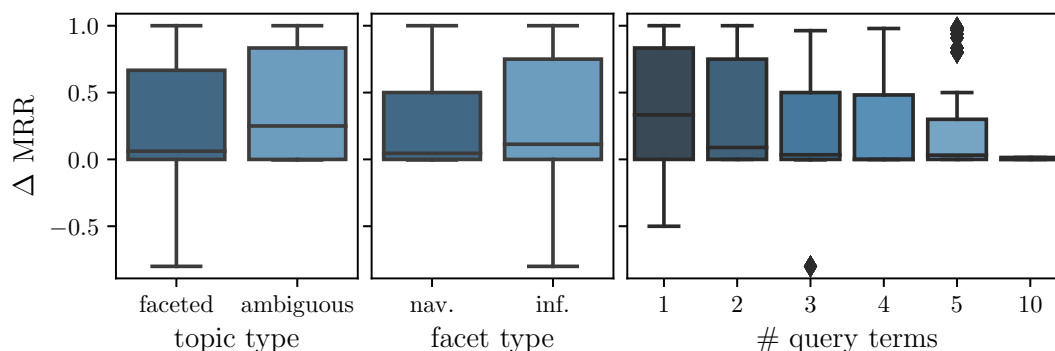


Figure 8.7. Impact of topic type, facet type, and query length on the performance of BestQuestion oracle model, compared to OriginalQuery.

tioning that we observe the highest relative improvements in terms of nDCG@1 (=173%) and P@1 (=158%), exhibiting a high potential impact on voice-only conversational systems.

Impact of topic type and length. We analyze the performance of BestQuestion based on the number of query terms and topic type. We see that the relative improvement of BestQuestion is negatively correlated with the number of query terms (Pearson’s $r = -0.2$, $p \ll 0.001$), suggesting that shorter queries require clarification in most cases. Also, comparing the topic types (ambiguous vs. faceted), we see a significant difference in the relative improvement. The average Δ MRR for ambiguous topics is 0.3858, compared with the faceted topics with average Δ MRR of 0.2898. The difference was statistically significant (2-way ANOVA, $p \ll 0.001$).

8.6.3 Question Selection

Table 8.3 presents the results of the document retrieval model taking into account a selected question together with its answer. We see that all models outperform OriginalQuery, confirming that asking clarifying questions is crucial in a conversation, and leading to high performance gain. For instance, compared to OriginalQuery, a model as simple as σ -QPP achieves a 31% relative improvement in terms of MRR. Also, NeuQS consistently outperforms all the baselines in terms of all evaluation metrics on both data splits. All the improvements are statistically significant. Moreover, NeuQS achieves a remarkable improvement in terms of both P@1 and nDCG@1. These two evaluation metrics are particularly important for voice-only conversational systems where the system must return only one result to the user. The obtained improvements highlight the necessity

and effectiveness of asking clarifying questions in a conversational search system, where they are perceived as natural means of interactions with users.

8.6.4 Impact of Data Splits

We compare the performance of models on both Qulac-T and Qulac-F data splits. We see that the LTR baselines perform worse on Qulac-F. Notice that the performance difference of LambdaMART among the splits is statistically significant in terms of all evaluation metrics ($p < 0.001$). RankNet, on the other hand, exhibits a more robust performance, i.e., the difference of its performance on the two data splits is not statistically significant. Unlike the baselines, NeuQS exhibits a significant improvement in terms of all evaluation metrics on Qulac-F ($p < 0.05$), except for nDCG@5. This suggests that the baseline models are prone to overfitting on queries and conversations in the training data. As mentioned, Qulac-F's train and test sets may have some queries and questions in common, hurting models that are weak at generalization.

8.6.5 Impact of Number of Conversation Turns

Figure 8.8 shows the performance of NeuQS as well as the baselines for different conversation turns. We evaluate different models at k turns ($k \in \{1, 2, 3\}$). We see that the performance of all models improves as the conversation advances to multiple turns. It is worth noting that a probabilistic modeling of users' interactions under the probabilistic framework proposed by Fuhr [85] to better understand the additional cost and effect that a system gets at each turn. Also, we see that all models consistently outperform the OriginalQuery baseline at different number of turns. Finally, we see that NeuQS exhibits robust performance, outperforming all the baselines at different turns.

8.6.6 Impact of Clarifying Questions on Facets

We study the difference of MRR between NeuQS and OriginalQuery on all facets. Note that for every facet we average the performance of NeuQS at different conversation turns. Our goal is to see how many facets are impacted positively by asking clarifying questions. NeuQS improves the effectiveness of retrieval by selecting relevant questions for a considerable number of facets on both data splits. In particular, the performance for 45% of the facets is improved by asking clarifying questions, whereas the performance for 19% becomes worse.

Table 8.3. Performance comparison with baselines on Qulac-T and Qulac-F.

Method	Qulac-T Dataset				Qulac-F Dataset					
	MRR	P@1	nDCG@1	nDCG@5	MRR	P@1	nDCG@1	nDCG@5	nDCG@20	
OriginalQuery	0.2715	0.1842	0.1381	0.1451	0.2715	0.1842	0.1381	0.1451	0.1470	
σ -QPP	0.3570	0.2548	0.1960	0.1938	0.3570	0.2548	0.1960	0.1938	0.1812	
LambdaMART	0.3558	0.2537	0.1945	0.1940	0.3501	0.2478	0.1911	0.1896	0.1773	
RankNet	0.3573	0.2562	0.1979	0.1943	0.3568	0.2559	0.1986	0.1944	0.1809	
NeuQS	0.3625*	0.2664*	0.2064*	0.2013*	0.1862*	0.3641*	0.2682*	0.2110*	0.2018*	0.1867*
WorstQuestion	0.2479	0.1451	0.1075	0.1402	0.2479	0.1451	0.1075	0.1402	0.1483	
BestQuestion	0.4673	0.3815	0.3031	0.2410	0.4673	0.3815	0.3031	0.2410	0.2077	

WorstQuestion and *BestQuestion* respectively determine the lower and upper bounds.

The superscript * denotes statistically significant differences compared to all the baselines ($p < 0.001$).

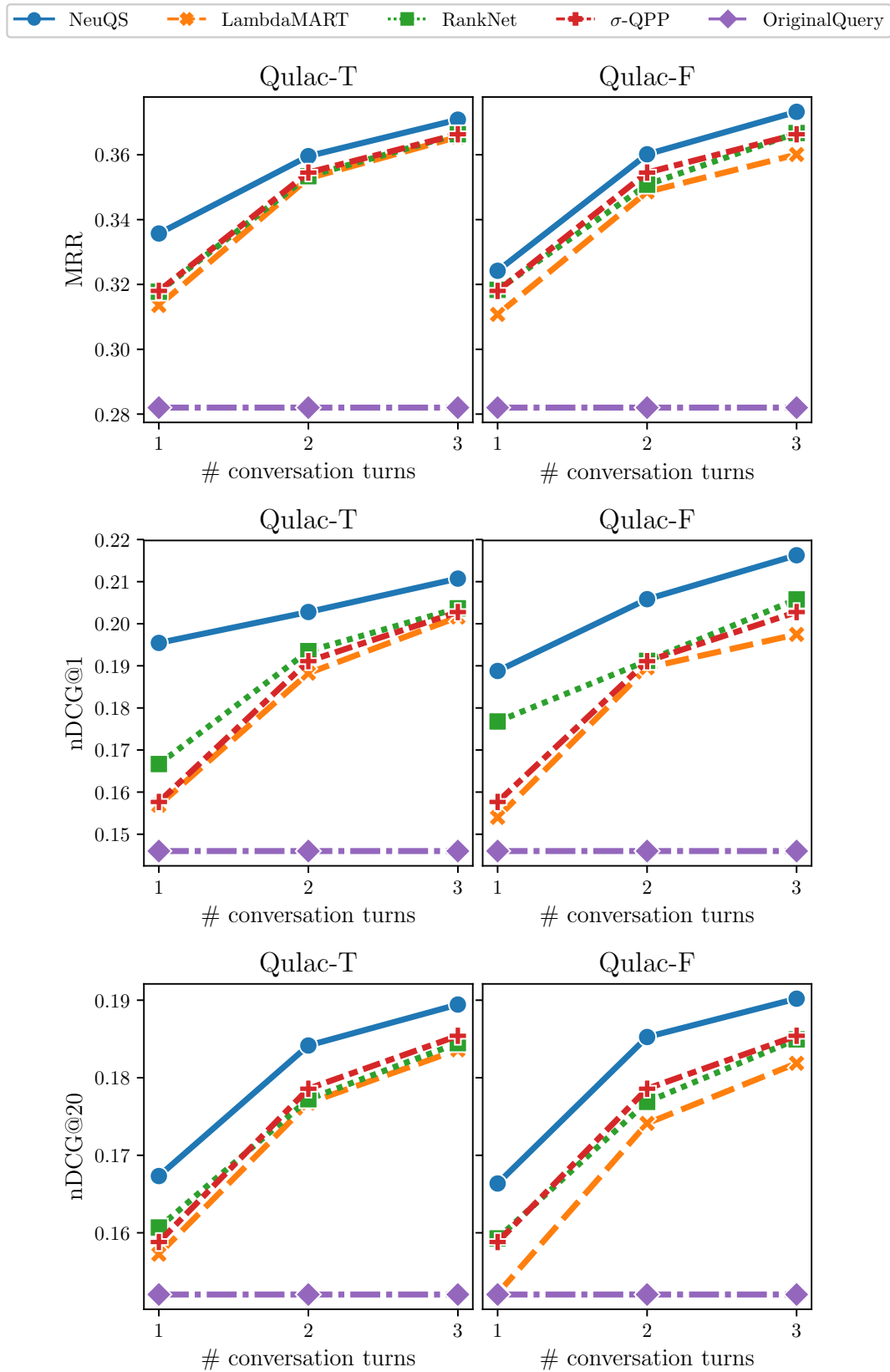


Figure 8.8. Performance comparison with the baselines for different number of conversation turns ($k \in \{1, 2, 3\}$).

Table 8.4. Failure and success examples of NeuQS. Failure and success are measured by the difference in performance of NeuQS and OriginalQuery in terms of MRR (Δ MRR).

Query	Facet Description	Selected Question	User's Answer	Δ MRR
dog heat	What is the effect of excessive heat on dogs?	Would you like to know how to care for your dog during heat?	No, I want to know what happens when a dog is too hot.	-0.86
sit and reach test	How is the sit and reach test properly done?	Do you want to know how to perform this test?	Yes, I do.	-0.75
alexian brothers hospital	Find Alexian Brothers hospitals.	Are you looking for our schedule of classes or events?	No, I don't need that.	-0.54
east ridge high school	Information about sports program at East Ridge High School in Clermont, Florida	What information about East Ridge High School are you looking for?	I'm looking for information about their sports program.	+0.96
euclid	Find information on the Greek mathematician Euclid.	Do you want a biography?	Yes.	+0.93
rocky mountain news	Who are the sports reporters for the Rocky Mountain News?	Would you like to read recent news about the Rocky Mountain News?	No, I just want a list of the reporters who write the sports for the Rocky Mountain News.	+0.88

8.6.7 Case Study: Failure and Success Analysis

Finally, in Table 8.4 we analyze representative cases of failure and success of our proposed framework. We list three cases where selecting questions using NeuQS improves the retrieval performance, as well as three other examples in which the selected questions lead to decreased performance. ΔMRR reports the difference of the performance of NeuQS and OriginalQuery in terms of MRR. As we see, the first three examples show the selected questions that hurt the performance (i.e., $\Delta\text{MRR} < 0.0$). The first row is an example where the user’s response to the question is negative; however, the user provides additional information about their information need (i.e., facet). We see that even though the user has provided additional information, the performance drops. This is perhaps due to lack of common terms between the additional information (i.e., “dog is too hot”) and the facet (i.e., “*excessive heat on dogs*”). This is more evident when we compare this example with a successful answer: “No, I would like to know the effects of *excessive heat on dogs*.”

The second row of the table shows a case where the answer to the question is positive, but there are no common terms between the question and the facet. Again, the intuition here is that the retrieval model is not able to take advantage of the additional information when it has no terms in common with the relevant documents.

The third row of the table shows another failure example where the selected question is not relevant to the facet and the user provides no additional information. This is a typical failure example where the system does not get any positive feedback, but could still use the negative feedback to improve the ranking. This can be done by diverging from the documents that are similar to the negative feedback.

As for the success examples, we have listed three types. The first example (“east ridge high school”) is where the system is able to ask an open question. Open questions are very hard to formulate for open-domain information-seeking scenarios; however, it is more likely to get useful feedback from users in response to such questions.

The fifth row shows an example of a positive feedback. The performance gain, in this case, is perhaps due to the existence of term “biography” in the question which would match with relevant documents. It is worth noting that the question and the query in this example have no common terms. This highlights the importance of employing a language-representation-based question retrieval model (e.g., BERT-LeaQuR) as opposed to term-matching IR models.

The last example shows a case where the answer is negative, but the user

is engaged in the conversation and provides additional information about the facet. We see that the answer contains keywords of the facet description (i.e., “reporters,” “sports”), improving the score of relevant documents that contain those terms.

8.7 Limitations

Every data collection comes with some limitations. The same is valid for Qulac. First, the dataset was not collected from actual conversations. This decision was mainly due to the unbalanced workload of the two conversation participants. In our crowdsourcing HITs, the task of question generation required nearly 10 times more effort compared to the task of question answering. This makes it challenging and more expensive to pair two workers as participants of the same conversation. There are some examples of this approach in the literature [59, 163]; however, they address the task of reading comprehension, a task that is considerably simpler than identifying topic facets. A possible future direction is to provide a limited number of pre-generated questions (say 10) to the workers to select from, so that the complexity of the task would be significantly reduced.

Furthermore, Qulac is built for single-turn conversations (i.e., one question; one answer). Even though there are questions that can be asked after one another to form a multi-turn conversation, our data collection approach does not guarantee the existence of multi-turn conversations that involve the same participants. Also, we believe that the quality of generated clarifying questions highly depends on how well the selected commercial search engine is able to diversify the result list. We aimed to minimize this bias by asking workers to scan at least three pages of the result list. Also, the questions added by expert annotators guarantees the coverage of all facets (see Section 8.3.3).

Finally, as we mentioned, faceted and ambiguous queries are good examples of topics that a conversational system needs to clarify; however, this task cannot be limited only to such queries. One can collect similar data for exploratory search scenarios, where asking questions can potentially lead to more user engagement while doing exploratory search.

In this chapter, our main focus was on question selection. There are various directions that can be explored in the future. One interesting problem is to explore various strategies for improving the performance of the document retrieval model as new information is added to the model. Moreover, we assumed the number of conversation turns to be fixed. Another interesting future direction is to model the system’s confidence at every stage of the conversation so that the

model is able to decide when to stop asking questions and present the result(s).

8.8 Summary

In this chapter, we introduced the task of asking clarifying questions in open-domain information-seeking conversations. We proposed an evaluation methodology which enables offline evaluation of conversational systems with clarifying questions. Also, we constructed and released a new data collection called Qulac, consisting of 762 topic-facet pairs with over 10K question-answer pairs. We further presented a neural question selection model called NeuQS along with models on question and document retrieval. NeuQS was able to significantly outperform the LTR baselines. The experimental analysis provided many insights of the task. In particular, experiments on the oracle model demonstrated that asking only one good clarifying question leads to over 150% relative improvement in terms of P@1 and nDCG@1. Moreover, we observed that asking clarifying questions improves the model's performance for a substantial percentage of the facets, despite the fact that a more effective retrieval model than the one we used could potentially improve the performance. Finally, we showed that, asking more clarifying questions leads to better results, once again confirming the effectiveness of asking clarifying questions in a conversational search system.

Chapter 9

Conclusions

9.1 Summary of the Work Carried Out

The primary motivation of this thesis was to address various aspects of user modeling on mobile devices. To this aim, we focused on three different aspects of mobile information access, that is, recommendation, unified search, and conversational search. The first step in modeling user information need is to build a personalized system, able to model users' past behavior and interests. To address this problem, we proposed enriching the user profiles for venue suggestion based on the history of preferences. Furthermore, we improved the user and venue profiles by incorporating complex contextual information such as trip type and accompanying people. Due to the known limitations of content-based recommender models, we also studied collaborative user modeling for venue suggestion. In particular, we proposed to incorporate geographical and temporal influences into a two-phase collaborative ranking model. The two-phase design of the algorithm enabled us to infer the user's preferences from their implicit feedback, i.e., check-ins. In the final chapter of the first part, we extended the collaborative framework to include an arbitrary number of venue-venue similarities to address the sparsity problem. Furthermore, we combined the proposed model with our content-based approach to introduce a hybrid model for venue suggestion, outperforming both content-based and collaborative state-of-the-art models.

As the users are nowadays exposed to a variety of mobile applications, each of which is concerned with a different type of information, featuring its search engine, in Part 2 we focused on the task of unified mobile search for the first time. We first introduced the task, highlighting its differences with distributed IR and federated search. Then, we ran two crowdsourcing tasks to understand

how people seek information on their devices, among various applications. The findings of our first study suggested that users tend to find information in various applications, with the majority of queries targeted to specific applications. We further proposed two neural app selection models to select target apps for given queries as the first step of building a unified mobile search system. In our second study, we took this work one step further and collected more realistic data of cross-app search queries. We asked the participants to report their everyday mobile queries, as well as the applications in which they issued the queries. The findings of our second study were in line with the first one and confirmed that users tend to seek information in various applications. It also shed light on other aspects of unified mobile search such as users temporal behavior and the effect of context.

In the last part of the thesis, we focused on conversational search and in particular, on asking clarifying questions in a conversation. We argued that asking clarifying questions is an essential part of a conversational search system as it enables the system to understand the users' information need more clearly. Moreover, since conversational systems can often return a limited number of results (if not just one), it is crucial for the system to clarify the user's information need before it is confident enough to return the result(s) to the user. To understand the effect of clarifying questions, we collected a dataset of questions and answers for 762 search scenarios where a single query could be interpreted in 3.85 different ways on average. The experiments showed that asking only one good clarifying question can lead up to over 150% relative improvement in terms of nDCG@1 and P@1, proving the importance and significance of clarifying questions in a conversational search system. Moreover, we proposed a retrieval framework consisting of question retrieval and selection, as well as document retrieval models.

The rest of this chapter is organized as follows. In Section 9.2 we revisit our contributions and provide a summary about them. Finally, in Section 9.3 we list the future research directions following from this thesis.

9.2 Main Contributions

In Part 1, we focused on the task of venue suggestion and, in particular, our contributions consisted in modeling users' information needs based on three approaches, namely, content-based, collaborative, and hybrid. The main contributions of Part 2 can be summarized as follows:

- We performed a personalized dimensionality reduction on venue taste keywords to address the sparsity problem for recommendation. To this aim, we

presented a probabilistic generative model that finds a statistical mapping between location taste keywords and user tags. We further elaborated on how we employed the EM algorithm to learn the parameters of the model.

- We studied how appropriateness information collected from a crowdsourcing platform can improve the effectiveness of recommendation by taking into account the appropriateness of a venue to the user's context. To this end, we collected and released two datasets containing contextual appropriateness features and labels.
- We proposed a novel two-phase CR-based POI recommendation algorithm incorporating users implicit check-in feedback with a focus on the top of the list. The two-phase design of the algorithm allowed us to follow a two-step strategy for incorporating the user's implicit feedback while regularizing the model using our proposed time-sensitive regularizer.

In Part 2, we presented and focused on the problem of a unified mobile search framework. The main contributions of Part 2 are summarized as follows:

- We introduced the task of unified search in mobile environments and discussed its differences with distributed IR and federated search, highlighting the reasons why it needs to be studied as a field. We then designed a crowdsourcing task in which we collected real-life mobile search tasks on various topics. We used the collected mobile search tasks to collect mobile search queries paired the mobile apps in which users would issue them.
- We presented the first study of user behavior while searching with different apps, as well as their search queries. We studied various attributes of the search queries that are submitted to different apps such as query terms and overlap. We also conducted an in-depth analysis of user behavior in terms of the apps they chose to complete a search task both individually and collectively.
- In an attempt to conduct a more realistic study and collect an in situ dataset, we designed and conducted a field study for collecting thousands of real-life cross-app queries. To this aim, we developed a bespoke Android app and asked the participants to install the app and let it run in the background for at least 24 hours. Hence, we collected the first in situ dataset of cross-app queries.

- We presented the first analysis of in situ cross-app queries and users' behavior as they search with different apps. More specifically, we studied different attributes of cross-app mobile queries concerning their target apps, sessions, and contexts. This study not confirmed the findings of our first study, but also provided several useful insights into how users access information via different apps to satisfy their everyday information needs.

In the last part of the thesis, we provided the first study on asking clarifying questions in open-domain information-seeking conversations. Our contributions in this part can be summarized as follows:

- We formulated the task of selecting and asking clarifying questions in open-domain information-seeking conversational systems. To this end, we proposed a sample workflow of such a system where the system can proactively ask clarifying questions.
- We proposed an offline evaluation framework for conversational systems with clarifying questions. We supported our proposed evaluation framework by collecting a dataset on top of the TREC Web Track 2009-2012 collections. Our dataset takes advantage of multi-faceted queries and relevance assessments that were initially targeted to evaluate search result diversification.
- We proposed a retrieval framework, consisting of three main components:
 1. Question retrieval aims to retrieve all relevant questions to a given query from the large pool of questions that is available in the dataset. The main goal in this component is to maximize the recall. This task is challenging since many questions have no terms in common with the query. We proposed a model that fine-tunes the BERT pre-trained model for this task.
 2. Question selection is concerned with selecting only one question from a list of candidate questions. Since a conversational system should eventually ask to select only one question to ask, it is important to take into account the query and conversation context while choosing the next question to ask. Hence, the main goal here is to maximize precision. We proposed a neural approach that learns to select the next question given multiple features on query, question, and conversation context.

3. Document retrieval component is used to update the retrieved list of documents one a question is a selection, and the user has provided an answer to it. It is crucial to incorporate the query, conversation context, as well as the latest question and answer into the ranking. Our proposed model is a simple linear interpolation of the QL model.
- We conducted oracle experiments and analyzed the behavior of the retrieval model under various conditions. We found that asking only one good question can lead up to 150% relative improvement in terms of P@1 and nDCG@1. Moreover, we observed asking clarifying questions have different impacts on different types of queries. In particular, ambiguous topics benefit more than faceted ones, informational facets helped more than navigational ones, and shorter queries benefited more than longer ones.
 - We generated artificial multi-turn conversations and studied the impact of multiple clarifying questions on the performance. We observed that our model, as well as all baselines, exhibited an improved performance as the conversation advanced.

9.3 Future Research Directions

We started the thesis by tackling a well-established problem in IR, that is, venue suggestion and provided new techniques and perspectives on this problem. Further, we studied a new task in mobile IR by collecting new datasets and analyzing them, followed by providing a new evaluation framework in conversational search. The trajectory of the thesis has stemmed specific future research directions on different topics. Below, we describe some of the possible future directions.

Cross-platform CR recommendation. A wealth of information about users and POIs are available on various LBSNs. As future work, we plan to extend our two-phase model to generate POI recommendations considering users' behavior in different LBSNs [7, 155]. Previous work has shown that considering a cross-platform and multimodal behavioral analysis improves the performance of a model dramatically. Hence, we are very interested in investigating how we can extend our current work to consider multimodal information such as POI category, user reviews, and opening hours. It would also be important to incorporate the sentiment of users in recommendation for such tasks [31, 33]

Recurrent neural networks for recommendation. Also, with the recent advances in applying deep neural models for POI recommendation [82, 127, 206] and their power to capture complicated structures of user-POI interactions, we plan to combine our joint learning approach with the existing deep recurrent neural models to explore its potential benefits to a deep neural recommender model.

Negative sampling for recommendation. Furthermore, since our training strategy requires negative training examples, we considered all unseen POIs as negative samples, which increases the complexity of the model. Inspired by the relevant studies [55, 201], we plan to explore various strategies for negative sampling and evaluate their effect on JTCR.

Multiple similarity functions for hybrid recommendation. We also plan to study how other similarity features, such as personal tags, time-based similarity, and contextual information from multiple LBSNs can improve venue suggestion and analyze their impact on our hybrid recommendation model.

Incorporating contextual information for target apps selection. Immediate future work for Part 2 can be exploring the influence of other types of contextual information, such as location and time, on the target apps selection task. As we saw in the analysis, temporal information shows a significant impact on the user's behavior. Therefore, it would be interesting to see how it can improve a target apps selection model.

Real-life unified mobile search. As described, our Android app is only used as a self-report data collection apparatus that is also able to collect sensor and contextual data. Therefore, our collected dataset does not reflect the actual behavior of users as they would search using a unified mobile search. It happens because no such system exists, and hence, one future direction is to develop such a system that can provide a unified interface for users' queries and retrieve results from multiple applications. Such a system would enable researchers to collect a more realistic dataset and uncover new aspects of this task to both industrial and research community.

Studies on results aggregation and presentation for unified mobile search. Also, it would be interesting to explore result aggregation and presentation in the future, considering two crucial factors: information gain and user satisfaction. This direction can be studied in both areas of information retrieval and human-computer interaction. It is essential to see whether users prefer to see an aggregated list of results (similar to the work done in federated search), or if they prefer to see results grouped based on the source apps.

Proactive information retrieval using cross-app queries. Based on our findings in the analyses, we believe that mobile search queries can be leveraged to improve the user experience. For instance, a user searches for a restaurant using a unified search system and finds some relevant information on Yelp. In this case, considering the user's personal preference as well as the context, the system could push notification with information about the traffic near the restaurant.

Studying the effect of difficult queries on system performance for unified mobile and conversational search. Query performance prediction and the effect of difficult queries have been studied extensively in the community [52, 58, 138]. Following a similar approach, we plan to understand how users and systems perform differently more difficult queries, as opposed to easier ones. More specifically, it would be interesting to see how clarifying questions would help a system for difficult queries, compared to easier ones. Also, we are interested in understanding whether users' perception of difficulty has any effect on their choice of app in a unified mobile search environment.

More realistic setting for collecting clarifying questions. We did not collect Qulac via actual conversations. This decision was mainly due to the unbalanced workload of the two conversation participants. In our crowdsourcing HITs, the task of question generation required nearly ten times more effort compared to the task of question answering. This makes it challenging and more expensive to pair two workers as participants of the same conversation. There are some examples of this approach in the literature [59, 163]; however, they address the task of reading comprehension, a task that is considerably simpler than identifying topic facets. A possible future direction is to provide a limited number of pre-generated questions (say 10) to the workers to select from, so that the complexity of the task would be significantly reduced.

Multi-turn conversations. Furthermore, Qulac is built for single-turn conversations (i.e., one question; one answer). Even though there are questions that can be asked after one another to form a multi-turn conversation, our data collection approach does not guarantee the existence of multi-turn conversations that involve the same participants. A possible future direction is to set up a crowdsourcing task similar to the one that we described in the previous future direction and allow participants to continue their conversation for multiple turns.

Clarifying questions for exploratory search. Faceted and ambiguous queries are good examples of topics that a conversational system needs to clarify; however, this task cannot be limited only to such queries. One can collect similar data for exploratory search scenarios, where asking questions can potentially lead to

more user engagement while doing an exploratory search.

Extended document retrieval model. In this work, our primary focus was on question selection. Various directions can be explored in the future. One interesting problem is to explore multiple strategies of improving the performance of the document retrieval model as new information is added to the model.

Modeling system confidence. Moreover, we assumed the number of conversation turns to be fixed. Another exciting future direction is to model the system's confidence at every stage of the conversation so that the model can decide when to stop asking questions and present the result(s).

Bibliography

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In *Recommender Systems Handbook*, pages 217–253. Springer, 2011.
- [2] Gediminas Adomavicius, Ramesh Sankaranarayanan, Shahana Sen, and Alexander Tuzhilin. Incorporating contextual information in recommender systems using a multidimensional approach. *ACM Trans. Inf. Syst.*, 23(1):103–145, 2005.
- [3] Charu C. Aggarwal. *Recommender Systems - The Textbook*. Springer, 2016. ISBN 978-3-319-29657-9.
- [4] Mohammad Aliannejadi and Fabio Crestani. Venue appropriateness prediction for personalized context-aware venue suggestion. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1177–1180. ACM, 2017.
- [5] Mohammad Aliannejadi and Fabio Crestani. A collaborative ranking model with contextual similarities for venue suggestion. In *Proceedings of the Italian Information Retrieval Workshop (IIR)*, 2018.
- [6] Mohammad Aliannejadi and Fabio Crestani. Venue suggestion using social-centric scores. *CoRR*, abs/1803.08354, 2018.
- [7] Mohammad Aliannejadi and Fabio Crestani. Personalized context-aware point of interest recommendation. *ACM Trans. Inf. Syst.*, 36(4):45:1–45:28, 2018.
- [8] Mohammad Aliannejadi, Shahram Khadivi, Saeed Shiry Ghidary, and Mohammad Hadi Bokaei. Discriminative spoken language understanding using statistical machine translation alignment models. In *International Symposium on Artificial Intelligence and Signal Processing*, pages 194–202. Springer, 2013.

- [9] Mohammad Aliannejadi, Masoud Kiaeeha, Shahram Khadivi, and Saeed Shiry Ghidary. Graph-based semi-supervised conditional random fields for spoken language understanding using unaligned data. In *Proceedings of the Australasian Language Technology Association Workshop (ALTA)*, pages 98–103, 2014.
- [10] Mohammad Aliannejadi, Seyed Ali Bahrainian, Anastasia Giachanou, and Fabio Crestani. University of Lugano at TREC 2015: Contextual suggestion and temporal summarization tracks. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2015.
- [11] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. User model enrichment for venue recommendation. In *Proceedings of the Asia Information Retrieval Societies Conference (AIRS)*, pages 212–223. Springer, 2016.
- [12] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. Venue appropriateness prediction for contextual suggestion. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2016.
- [13] Mohammad Aliannejadi, Maram Hasanain, Jiaxin Mao, Jaspreet Singh, Johanne R. Trippas, Hamed Zamani, and Laura Dietz. ACM SIGIR student liaison program. *SIGIR Forum*, 51(3):42–45, 2017.
- [14] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. Personalized ranking for context-aware venue suggestion. In *Proceedings of the Symposium on Applied Computing (SAC)*, pages 960–962. ACM, 2017.
- [15] Mohammad Aliannejadi, Ida Mele, and Fabio Crestani. A cross-platform collection for contextual suggestion. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1269–1272. ACM, 2017.
- [16] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. Personalized keyword boosting for venue suggestion based on multiple lbsns. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 291–303. Springer, 2017.
- [17] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. A collaborative ranking model with multiple location-based similarities for venue suggestion. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 19–26. ACM, 2018.

- [18] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. In situ and context-aware target apps selection for unified mobile search. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1383–1392. ACM, 2018.
- [19] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Target apps selection: Towards a unified search framework for mobile devices. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 215–224. ACM, 2018.
- [20] Mohammad Aliannejadi, Morgan Harvey, Luca Costa, Matthew Pointon, and Fabio Crestani. Understanding mobile search task relevance and user behaviour in context. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 143–151. ACM, 2019.
- [21] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 475–484. ACM, 2019.
- [22] Mohammad Aliannejadi, Dimitrios Rafailidis, and Fabio Crestani. A joint two-phase time-sensitive regularized collaborative ranking model for point of interest recommendation. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 2019 (in press).
- [23] Omar Alonso and Maria Stone. Building a query log via crowdsourcing. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 939–942. ACM, 2014.
- [24] Giambattista Amati. *Probability models for information retrieval based on divergence from randomness*. PhD thesis, University of Glasgow, UK, 2003.
- [25] Avi Arampatzis and Georgios Kalamatianos. Suggesting points-of-interest via content-based, collaborative, and hybrid fusion methods in mobile devices. *ACM Trans. Inf. Syst.*, 36(3):23:1–23:28, 2017.
- [26] Jaime Arguello. Aggregated search. *Foundations and Trends in Information Retrieval*, 10(5):365–502, 2017.

- [27] Jaime Arguello, Jamie Callan, and Fernando Diaz. Classification-based resource selection. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1277–1286. ACM, 2009.
- [28] Harald Aust, Martin Oerder, Frank Seide, and Volker Steinbiss. The philips automatic train timetable information system. *Speech Communication*, 17(3-4):249–262, 1995.
- [29] Seyed Ali Bahrainian and Fabio Crestani. Towards the next generation of personal assistants: Systems that know when you forget. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval, ICTIR*, pages 169–176, 2017.
- [30] Seyed Ali Bahrainian and Fabio Crestani. Augmentation of human memory: Anticipating topics that continue in the next meeting. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 150–159. ACM, 2018.
- [31] Seyed Ali Bahrainian and Andreas Dengel. Sentiment analysis of texts by capturing underlying sentiment patterns. *Web Intelligence*, 13(1):53–68, 2015.
- [32] Seyed Ali Bahrainian, Seyed Mohammad Bahrainian, Meytham Salarnasab, and Andreas Dengel. Implementation of an intelligent product recommender system in an e-store. In *Proceedings Active Media Technology, 6th International Conference, AMT*, pages 174–182, 2010.
- [33] Seyed Ali Bahrainian, Marcus Liwicki, and Andreas Dengel. Fuzzy subjective sentiment phrases: A context sensitive and self-maintaining sentiment lexicon. In *2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT)*, pages 361–368, 2014.
- [34] Seyed Ali Bahrainian, Ida Mele, and Fabio Crestani. Modeling discrete dynamic topics. In *Proceedings of the Symposium on Applied Computing, SAC*, pages 858–865, 2017.
- [35] Seyed Ali Bahrainian, Ida Mele, and Fabio Crestani. Predicting topics in scholarly papers. In *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR*, pages 16–28, 2018.

- [36] Suhrid Balakrishnan and Sumit Chopra. Collaborative ranking. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 143–152, 2012.
- [37] Nicholas J Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications*, 9(3):379–395, 1995.
- [38] Jan R. Benetka, Krisztian Balog, and Kjetil Nørkvåg. Anticipating information needs based on check-in activity. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 41–50. ACM, 2017.
- [39] Paul N. Bennett, Filip Radlinski, Ryen W. White, and Emine Yilmaz. Inferring and using location metadata to personalize web search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 135–144. ACM, 2011.
- [40] Agon Bexheti, Evangelos Niforatos, Seyed Ali Bahrainian, Marc Langheinrich, and Fabio Crestani. Measuring the effect of cued recall on work meetings. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1020–1026, 2016.
- [41] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3:993–1022, 2003.
- [42] Hamed Bonab, Mohammad Aliannejadi, John Foley, and James Allan. Incorporating hierarchical domain information to disambiguate very short queries. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 51–54. ACM, 2019.
- [43] Joel Brandt, Noah Weiss, and Scott R. Klemmer. txt 4 l8r: lowering the burden for diary studies under mobile conditions. In *Extended Abstracts Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, pages 2303–2308. ACM, 2007.
- [44] Pavel Braslavski, Denis Savenkov, Eugene Agichtein, and Alina Dubatovka. What do you mean exactly?: Analyzing clarification questions in CQA. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 345–348. ACM, 2017.

- [45] Matthias Braunhofer, Mehdi Elahi, and Francesco Ricci. Techniques for cold-starting context-aware mobile recommender systems for tourism. *Intelligenza Artificiale*, 8(2):129–143, 2014.
- [46] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 89–96. PMLR, 2005.
- [47] Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 89–96, Bonn, Germany, 2005. PMLR.
- [48] James P. Callan and Margaret E. Connell. Query-based sampling of text databases. *ACM Trans. Inf. Syst.*, 19(2):97–130, 2001.
- [49] Huanhuan Cao, Derek Hao Hu, Dou Shen, Daxin Jiang, Jian-Tao Sun, Enhong Chen, and Qiang Yang. Context-aware query classification. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3–10. ACM, 2009.
- [50] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning (ICML)*. PMLR, 2007.
- [51] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 129–136. PMLR, 2007.
- [52] Claudio Carpineto, Stefano Mizzaro, Giovanni Romano, and Matteo Snidero. Mobile information retrieval with search results clustering: Prototypes and evaluations. *JASIST*, 60(5):877–895, 2009.
- [53] Juan Pablo Carrascal and Karen Church. An in-situ study of mobile app & mobile search interactions. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, pages 2739–2748, 2015.
- [54] Li Chen, Guanliang Chen, and Feng Wang. Recommender systems based on user reviews: the state of the art. *User Model. User-Adapt. Interact.*, 25(2):99–154, 2015.

- [55] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. Fused matrix factorization with geographical and social influence in location-based social networks. In Jörg Hoffmann and Bart Selman, editors, *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 17–23. AAAI Press, 2012.
- [56] Chen Cheng, Haiqin Yang, Michael R. Lyu, and Irwin King. Where you like to go next: Successive point-of-interest recommendation. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China*, pages 2605–2611, 2013.
- [57] Chen Cheng, Haiqin Yang, Irwin King, and Michael R. Lyu. A unified point-of-interest recommendation framework in location-based social networks. *ACM Trans. Intell. Syst. Technol.*, 8(1):10:1–10:21, 2016.
- [58] Adrian-Gabriel Chifu, Léa Laporte, Josiane Mothe, and Md Zia Ullah. Query performance prediction focused on summarized letor features. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR*, pages 1177–1180. ACM, 2018.
- [59] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac: Question answering in context. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2174–2184, 2018.
- [60] Konstantina Christakopoulou and Arindam Banerjee. Collaborative ranking with a push at the top. In *Proceedings of the 24th International Conference on World Wide Web, Florence, Italy*, pages 205–215, 2015.
- [61] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 815–824, 2016.
- [62] Karen Church and Nuria Oliver. Understanding mobile web and mobile search use in today’s dynamic mobile landscape. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 67–76. ACM, 2011.
- [63] Karen Church and Barry Smyth. Understanding the intent behind mobile information needs. In *Proceedings of the International Conference on Intelligent User Interfaces (IUI)*, pages 247–256. ACM, 2009.

- [64] Karen Church, Barry Smyth, Paul Cotter, and Keith Bradley. Mobile information access: A study of emerging search behavior on the mobile internet. *TWEB*, 1(1):4, 2007.
- [65] Karen Church, Barry Smyth, Keith Bradley, and Paul Cotter. A large scale study of european mobile search behaviour. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 13–22. ACM, 2008.
- [66] Karen Church, Mauro Cherubini, and Nuria Oliver. A large-scale study of daily information needs captured in situ. *ACM Trans. Comput.-Hum. Interact.*, 21(2):10:1–10:46, 2014.
- [67] Charles L. A. Clarke, Nick Craswell, and Ian Soboroff. Overview of the TREC 2009 web track. In *Proceedings of the Text REtrieval Conference (TREC)*, 2009.
- [68] Charles L. A. Clarke, Nick Craswell, Ian Soboroff, and Ellen M. Voorhees. Overview of the TREC 2011 web track. In *Proceedings of the Text REtrieval Conference (TREC)*, 2011.
- [69] Charles L. A. Clarke, Nick Craswell, and Ellen M. Voorhees. Overview of the TREC 2012 web track. In *Proceedings of the Text REtrieval Conference (TREC)*, 2012.
- [70] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.
- [71] Koby Crammer and Yoram Singer. Pranking with ranking. In *Proceedings of the Neural Information Processing Systems: Natural and Synthetic, Vancouver, British Columbia, Canada*, pages 641–647, 2001.
- [72] Fabio Crestani and Heather Du. Written versus spoken queries: A qualitative and quantitative comparative analysis. *JASIST*, 57(7):881–890, 2006.
- [73] Fabio Crestani, Stefano Mizzaro, and Ivan Scagnetto. *Mobile Information Retrieval*. Springer Briefs in Computer Science. Springer, 2017.
- [74] W. Bruce Croft and R. H. Thompson. I³r: A new approach to the design of document retrieval systems. *JASIS*, 38(6):389–404, 1987.

- [75] Chaoran Cui, Jialie Shen, Liqiang Nie, Richang Hong, and Jun Ma. Augmented collaborative filtering for sparseness reduction in personalized POI recommendation. *ACM Trans. Intell. Sys. and Tech.*, 8(5):71:1–71:23, 2017.
- [76] Adriel Dean-Hall, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. Overview of the TREC 2015 contextual suggestion track. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2015.
- [77] Mostafa Dehghani, Hamed Zamani, Aliaksei Severyn, Jaap Kamps, and W. Bruce Croft. Neural ranking models with weak supervision. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 65–74. ACM, 2017.
- [78] Romain Deveaud, M-Dyaa Albakour, Craig Macdonald, and Iadh Ounis. Experiments with a venue-centric model for personalised and time-aware venue suggestion. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 53–62. ACM, 2015.
- [79] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.
- [80] Fernando Diaz. Integration of news content into web results. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 182–191. ACM, 2009.
- [81] Yi Ding and Xue Li. Time weight collaborative filtering. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 485–492. ACM, 2005.
- [82] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1555–1564, 2016.
- [83] Quan Fang, Changsheng Xu, M. Shamim Hossain, and Ghulam Muhammad. STCAPLRS: A spatial-temporal context-aware personalized location recommendation system. *ACM Trans. Intell. Syst. Technol.*, 7(4):59:1–59:30, 2016.

- [84] Gregory Ferenç, Mao Ye, and Wang-Chien Lee. Location recommendation for out-of-town users in location-based social networks. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 721–726. ACM, 2013.
- [85] Norbert Fuhr. A probability ranking principle for interactive information retrieval. *Inf. Retr.*, 11(3):251–265, 2008.
- [86] Huiji Gao, Jiliang Tang, Xia Hu, and Huan Liu. Exploring temporal effects for location recommendation on location-based social networks. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 93–100. ACM, 2013.
- [87] StatCounter GlobalStats. Mobile and tablet internet usage exceeds desktop for first time worldwide. <http://gs.statcounter.com/press/mobile-and-tablet-internet-usage-exceeds-desktop-for-first-time-worldwide>, 2016.
- [88] David Goldberg, David A. Nichols, Brian M. Oki, and Douglas B. Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, 1992.
- [89] Google and Nielsen. Mobile search moments: Understanding how mobile drives conversions. <https://www.thinkwithgoogle.com/consumer-insights/creating-moments-that-matter/>, 2013.
- [90] Jean-Benoît Griesner, Talel Abdesslem, and Hubert Naacke. POI recommendation: Towards fused matrix factorization with geographical and temporal influences. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 301–304. ACM, 2015.
- [91] Ido Guy. Searching by talking: Analysis of voice queries on mobile web search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 35–44. ACM, 2016.
- [92] Negar Hariri, Bamshad Mobasher, Robin Burke, and Yong Zheng. Context-aware recommendation based on review mining. In *Proceedings of the 9th Workshop on Intelligent Techniques for Web Personalization & Recommender Systems, ITWP@IJCAI*, volume 756. CEUR-WS.org, 2011.
- [93] Morgan Harvey and Matthew Pointon. Searching on the go: The effects of fragmented attention on mobile web search tasks. In *Proceedings of*

- the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 155–164. ACM, 2017.
- [94] Morgan Harvey and Matthew Pointon. Searching on the go: the effects of fragmented attention on mobile web search tasks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 155–164. ACM, 2017.
- [95] Helia Hashemi, Mohammad Aliannejadi, Hamed Zamani, and W. Bruce Croft. ANTIQUE: A non-factoid question answering benchmark. *CoRR*, abs/1905.08957, 2019.
- [96] Seyyed Hadi Hashemi, Charles L. A. Clarke, Jaap Kamps, Julia Kiseleva, and Ellen M. Voorhees. Overview of the TREC 2016 contextual suggestion track. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2016.
- [97] Shun Hattori, Taro Tezuka, and Katsumi Tanaka. Context-aware query refinement for mobile web search. In *SAINT Workshops*, 2007.
- [98] Yulan He and Steve J. Young. Semantic processing using the hidden vector state model. *Computer Speech & Language*, 19(1):85–106, 2005.
- [99] Charles T. Hemphill, John J. Godfrey, and George R. Doddington. The ATIS spoken language systems pilot corpus. In *HLT*, pages 96–101, 1990.
- [100] Jun Hu and Ping Li. Decoupled collaborative ranking. In Rick Barrett, Rick Cummings, Eugene Agichtein, and Evgeniy Gabrilovich, editors, *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1321–1329. ACM, 2017.
- [101] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 8th IEEE International Conference on Data Mining, Pisa, Italy*, pages 263–272, 2008.
- [102] Di Jiang, Kenneth Wai-Ting Leung, Lingxiao Yang, and Wilfred Ng. Query suggestion with diversification and personalization. *Knowl.-Based Syst.*, 89:553–568, 2015.
- [103] Thorsten Joachims, Dayne Freitag, and Tom M. Mitchell. Web watcher: A tour guide for the world wide web. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97, Nagoya, Japan, August 23-29, 1997, 2 Volumes*, pages 770–777. Morgan Kaufmann, 1997.

- [104] Anne Kaikkonen. Full or tailored mobile web-where and how do people browse on their mobiles? In *Proceedings of the 5th International Conference on Mobile Technology, Applications, and Systems, MobiSys '08, Yilan, Taiwan*, pages 28:1–28:8. ACM, 2008.
- [105] Maryam Kamvar and Shumeet Baluja. A large scale study of wireless search behavior: Google mobile search. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, pages 701–709, 2006.
- [106] Maryam Kamvar and Shumeet Baluja. The role of context in query input: using contextual signals to complete queries on mobile devices. In *Proceedings of the International Conference on Human Computer Interaction with Mobile Devices and Services (MobileHCI)*, pages 405–412, 2007.
- [107] In-Ho Kang and Gil-Chang Kim. Query type classification for web document retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 64–71. ACM, 2003.
- [108] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 79–86. ACM, 2010.
- [109] Makoto P. Kato and Katsumi Tanaka. To suggest, or not to suggest for queries with diverse intents: Optimizing search result presentation. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 133–142, 2016.
- [110] Abbas Keramati, Ruholla Jafari-Marandi, M. Aliannejadi, I. Ahmadian, M. Mozaffari, and U. Abbasi. Improved churn prediction in telecommunication industry using data mining techniques. *Appl. Soft Comput.*, 24: 994–1012, 2014.
- [111] Johannes Kiesel, Arefeh Bahrami, Benno Stein, Avishek Anand, and Matthias Hagen. Toward voice query clarification. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1257–1260. ACM, 2018.
- [112] Jaewon Kim, Paul Thomas, Ramesh Sankaranarayanan, Tom Gedeon, and Hwan-Jin Yoon. What snippet size is needed in mobile web search? In

- Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 97–106. ACM, 2017.
- [113] Aniket Kittur, Ed H. Chi, and Bongwon Suh. Crowdsourcing user studies with mechanical turk. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, pages 453–456, 2008.
- [114] Weize Kong and James Allan. Extracting query facets from search results. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 93–102. ACM, 2013.
- [115] Marijn Koolen, Toine Bogers, Antal van den Bosch, and Jaap Kamps. Looking for books in social media: An analysis of complex search requests. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 184–196, 2015.
- [116] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '08, Las Vegas, NV, USA*, pages 426–434, August 2008. doi: 10.1145/1401890.1401944.
- [117] Taku Kudo and Yuji Matsumoto. Fast methods for kernel-based text analysis. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 24–31. ACL, 2003.
- [118] John Lafferty and Chengxiang Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 111–119. ACM, 2001.
- [119] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 282–289. PMLR, 2001.
- [120] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 120–127. ACM, 2001.
- [121] Joonseok Lee, Samy Bengio, Seungyeon Kim, Guy Lebanon, and Yoram Singer. Local collaborative ranking. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 85–96. ACM, 2014.

- [122] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system demo. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 305–306. ACM, 2012.
- [123] Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing*. Morgan & Claypool Publishers, 2011. ISBN 1608457079, 9781608457076.
- [124] Xin Li, Mingming Jiang, Huiting Hong, and Lejian Liao. A time-aware personalized point-of-interest recommendation via high-order tensor factorization. *ACM Trans. Inf. Syst.*, 35(4):31:1–31:23, 2017.
- [125] Xutao Li, Gao Cong, Xiaoli Li, Tuan-Anh Nguyen Pham, and Shonali Krishnaswamy. Rank-geomf: A ranking based geographical factorization method for point of interest recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 433–442. ACM, 2015.
- [126] Defu Lian, Cong Zhao, Xing Xie, Guangzhong Sun, Enhong Chen, and Yong Rui. Geomf: joint geographical modeling and matrix factorization for point-of-interest recommendation. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 831–840, 2014.
- [127] Qiang Liu, Shu Wu, Liang Wang, and Tieniu Tan. Predicting the next location: A recurrent model with spatial and temporal contexts. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 194–200, 2016.
- [128] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.
- [129] Yanchi Liu, Chuanren Liu, Bin Liu, Meng Qu, and Hui Xiong. Unified point-of-interest recommendation with temporal interval assessment. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1015–1024. ACM, 2016.
- [130] Yiding Liu, Tuan-Anh Pham, Gao Cong, and Quan Yuan. An experimental evaluation of point-of-interest recommendation in location-based social networks. *PVLDB*, 10(10):1010–1021, 2017.

- [131] Yong Liu, Wei Wei, Aixin Sun, and Chunyan Miao. Exploiting geographical neighborhood characteristics for location recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 739–748. ACM, 2014.
- [132] Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor, editors, *Recommender Systems Handbook*, pages 73–105. Springer, 2011.
- [133] Xiaolu Lu, Alistair Moffat, and J. Shane Culpepper. The effect of pooling and evaluation depth on IR metrics. *Inf. Retr. Journal*, 19(4):416–445, 2016.
- [134] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.
- [135] Jiaxin Mao, Yiqun Liu, Noriko Kando, Cheng Luo, Min Zhang, and Shaoping Ma. Investigating result usefulness in mobile search. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 223–236. Springer, 2018.
- [136] Ida Mele, Seyed Ali Bahrainian, and Fabio Crestani. Linking news across multiple streams for timeliness analysis. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management, CIKM*, pages 767–776, 2017.
- [137] Prem Melville, Raymond J. Mooney, and Ramadass Nagarajan. Content-boosted collaborative filtering for improved recommendations. In Rina Dechter, Michael J. Kearns, and Richard S. Sutton, editors, *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 187–192. AAAI Press, 2002.
- [138] Stefano Mizzaro and Josiane Mothe. Why do you think this query is difficult?: A user study on human query prediction. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR*, pages 1073–1076. ACM, 2016.
- [139] George D. Montanez, Ryen W. White, and Xiao Huang. Cross-device search. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1669–1678. ACM, 2014.

- [140] Kevin Ong, Kalervo Järvelin, Mark Sanderson, and Falk Scholer. Using information scent to understand mobile and desktop web search behavior. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 295–304. ACM, 2017.
- [141] Iadh Ounis, Gianni Amati, Vassilis Plachouras, Ben He, Craig Macdonald, and Douglas Johnson. Terrier information retrieval platform. In *Proceedings of the European Conference on IR Research (ECIR)*, pages 517–519. Springer, 2005.
- [142] Harshith Padigela, Hamed Zamani, and W. Bruce Croft. Investigating the successes and failures of bert for passage re-ranking. *arXiv:1903.06902*, 2019.
- [143] Rong Pan, Yunhong Zhou, Bin Cao, Nathan Nan Liu, Rajan M. Lukose, Martin Scholz, and Qiang Yang. One-class collaborative filtering. In *Proceedings of the 8th IEEE International Conference on Data Mining, Pisa, Italy*, pages 502–511, 2008.
- [144] Dae Hoon Park, Mengwen Liu, ChengXiang Zhai, and Haohong Wang. Leveraging user reviews to improve accuracy for mobile app retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 533–542. ACM, 2015.
- [145] Dae Hoon Park, Yi Fang, Mengwen Liu, and ChengXiang Zhai. Mobile app retrieval for social media users via inference of implicit intent in social media text. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 959–968. ACM, 2016.
- [146] Moon-Hee Park, Jin-Hyuk Hong, and Sung-Bae Cho. Location-based recommendation system using bayesian user’s preference model in mobile devices. In *Proceedings of the 4th International Conference Ubiquitous Intelligence and Computing, UIC ’07, Hong Kong, China*, pages 1130–1139. Springer, 2007.
- [147] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014.
- [148] Joaquín Pérez-Iglesias and Lourdes Araujo. Standard deviation as a query hardness estimator. In *SPIRE*, pages 207–212, 2010.

- [149] Roberto Pieraccini, Evelyne Tzoukermann, Z. Gorelov, Jean-Luc Gauvain, Esther Levin, Chin-Hui Lee, and Jay Wilpon. A speech understanding system based on statistical representation of semantics. In *ICASSP*, pages 193–196, 1992.
- [150] Jay M. Ponte and W. Bruce Croft. A language modeling approach to information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 275–281. ACM, 1998.
- [151] Minghui Qiu, Liu Yang, Feng Ji, Wei Zhou, Jun Huang, Haiqing Chen, W. Bruce Croft, and Wei Lin. Transfer learning for context-aware question matching in information-seeking conversations in e-commerce. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (2)*, pages 208–213, 2018.
- [152] Chen Qu, Liu Yang, W. Bruce Croft, Johanne R. Trippas, Yongfeng Zhang, and Minghui Qiu. Analyzing and characterizing user intent in information-seeking conversations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 989–992. ACM, 2018.
- [153] Filip Radlinski and Nick Craswell. A theoretical framework for conversational search. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 117–126. ACM, 2017.
- [154] Dimitrios Rafailidis and Fabio Crestani. Joint collaborative ranking with social relationships in top-n recommendation. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1393–1402. ACM, 2016.
- [155] Dimitrios Rafailidis and Fabio Crestani. Top-n recommendation via joint cross-domain user clustering and similarity learning. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (PKDD)*, 2016.
- [156] Dimitrios Rafailidis and Fabio Crestani. Collaborative ranking with social relationships for top-n recommendations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 785–788. ACM, 2016.

- [157] Dimitrios Rafailidis and Fabio Crestani. Learning to rank with trust and distrust in recommender systems. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 5–13. ACM, 2017.
- [158] Hossein A. Rahmani, Mohammad Aliannejadi, Sajad Ahmadian, Mitra Baratchi, Mohsen Afsharchi, and Fabio Crestani. LGLMF: local geographical based logistic matrix factorization model for POI recommendation. In *Proceedings of the Asia Information Retrieval Societies Conference (AIRS)*, 2019.
- [159] Hossein A. Rahmani, Mohammad Aliannejadi, Rasoul Mirzaei Zadeh, Mitra Baratchi, Mohsen Afsharchi, and Fabio Crestani. Category-aware location embedding for point-of-interest recommendation. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 173–176. ACM, 2019.
- [160] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don’t know: Unanswerable questions for squad. *arXiv:1806.03822*, 2018.
- [161] Sudha Rao and Hal Daumé. Learning to ask good questions: Ranking clarification questions using neural expected value of perfect information. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (1)*, pages 2736–2745, 2018.
- [162] Sudha Rao and Hal Daumé III. Answer-based adversarial training for generating clarification questions. *arXiv:1904.02281*, 2019.
- [163] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge. *arXiv:1808.07042*, 2018.
- [164] Gary Ren, Xiaochuan Ni, Manish Malik, and Qifa Ke. Conversational query understanding using sequence to sequence modeling. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1715–1724, 2018.
- [165] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. In *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada*, pages 452–461, 2009.
- [166] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. Okapi at TREC-3. In *Proceedings of the Text REtrieval Conference (TREC)*, pages 109–126, 1994.

- [167] Cynthia Rudin. The p-norm push: A simple convex ranking algorithm that concentrates at the top of the list. *Journal of Machine Learning Research*, 10:2233–2271, 2009.
- [168] James Salter and Nick Antonopoulos. Cinemascreen recommender agent: Combining collaborative and content-based filtering. *IEEE Intelligent Systems*, 21(1):35–41, 2006.
- [169] Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference, Hong Kong, China*, pages 285–295, 2001.
- [170] Dou Shen, Jian-Tao Sun, Qiang Yang, and Zheng Chen. Building bridges for web query classification. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 131–138. ACM, 2006.
- [171] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 43–50. ACM, 2005.
- [172] Yue Shi, Alexandros Karatzoglou, Linas Baltrunas, Martha Larson, and Alan Hanjalic. Gapfm: optimal top-n recommendations for graded relevance domains. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 2261–2266. ACM, 2013.
- [173] Milad Shokouhi and Qi Guo. From queries to cards: Re-ranking proactive card recommendations based on reactive search history. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 695–704. ACM, 2015.
- [174] Milad Shokouhi, Rosie Jones, Umut Ozertem, Karthik Raghunathan, and Fernando Diaz. Mobile query reformulations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1011–1014. ACM, 2014.
- [175] Craig Silverstein, Monika Rauch Henzinger, Hannes Marais, and Michael Moricz. Analysis of a very large web search engine query log. *SIGIR Forum*, 33(1):6–12, 1999.

- [176] Timothy Sohn, Kevin A. Li, William G. Griswold, and James D. Hollan. A diary study of mobile information needs. In *Proceedings of the CHI Conference on Human Factors in Computing Systems (CHI)*, pages 433–442. ACM, 2008.
- [177] Yang Song, Hao Ma, Hongning Wang, and Kuansan Wang. Exploring and exploiting user search behavior on mobile and tablet devices to improve search relevance. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1201–1212, 2013.
- [178] Damiano Spina, Johanne R. Trippas, Lawrence Cavedon, and Mark Sanderson. Extracting audio summaries to support effective spoken document search. *JASIST*, 68(9):2101–2115, 2017.
- [179] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.
- [180] Yu Sun, Nicholas Jing Yuan, Xing Xie, Kieran McDonald, and Rui Zhang. Collaborative intent prediction with real-time contextual data. *ACM Trans. Inf. Syst.*, 35(4):30:1–30:33, 2017.
- [181] Yueming Sun and Yi Zhang. Conversational recommender system. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 235–244. ACM, 2018.
- [182] Zhiliang Tian, Rui Yan, Lili Mou, Yiping Song, Yansong Feng, and Dongyan Zhao. How to make context more useful? an empirical study on context-aware neural conversational models. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (2)*, pages 231–236, 2017.
- [183] Johanne R. Trippas, Damiano Spina, Lawrence Cavedon, Hideo Joho, and Mark Sanderson. Informing the design of spoken conversational search: Perspective paper. In *Proceedings of the Conference Human Information Interaction and Retrieval (CHIIR)*, pages 32–41. ACM, 2018.
- [184] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv:1706.03762*, 2017.

- [185] Alexandra Vtyurina, Denis Savenkov, Eugene Agichtein, and Charles L. A. Clarke. Exploring conversational search with humans, assistants, and wizards. In *CHI Extended Abstracts*, pages 2187–2193, 2017.
- [186] Marilyn A. Walker, Rebecca J. Passonneau, and Julie E. Boland. Quantitative and qualitative evaluation of darpa communicator spoken dialogue systems. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 515–522, 2001.
- [187] Yansen Wang, Chenyi Liu, Minlie Huang, and Liqiang Nie. Learning to ask questions in open-domain conversational systems with typed decoders. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL) (1)*, pages 2193–2203, 2018.
- [188] Ahmad M. Ahmad Wasfi. Collecting user access patterns for building user profiles and collaborative filtering. In Mark T. Maybury, Pedro A. Szekely, and Christoph G. Thomas, editors, *Proceedings of the 4th International Conference on Intelligent User Interfaces, IUI 1999, Los Angeles, CA, USA, January 5-8, 1999*, pages 57–64. ACM, 1999.
- [189] Markus Weimer, Alexandros Karatzoglou, Quoc V. Le, and Alexander J. Smola. COFI RANK - maximum margin matrix factorization for collaborative ranking. In *Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada*, pages 1593–1600, 2007.
- [190] Ryen W. White, Paul N. Bennett, and Susan T. Dumais. Predicting short-term interests using activity-based search context. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 1009–1018. ACM, 2010.
- [191] Jason D. Williams, Antoine Raux, Deepak Ramachandran, and Alan W. Black. The dialog state tracking challenge. In *SIGDIAL*, pages 404–413, 2013.
- [192] Kyle Williams, Julia Kiseleva, Aidan C. Crook, Imed Zitouni, Ahmed Hassan Awadallah, and Madian Khabsa. Detecting good abandonment in mobile search. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 495–505. ACM, 2016.

- [193] Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Inf. Retr. Journal*, 13(3):254–270, 2010.
- [194] Biao Xiang, Daxin Jiang, Jian Pei, Xiaohui Sun, Enhong Chen, and Hang Li. Context-aware ranking in web search. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 451–458. ACM, 2010.
- [195] Rui Yan, Yiping Song, and Hua Wu. Learning to respond with deep neural networks for retrieval-based human-computer conversation system. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 55–64. ACM, 2016.
- [196] Rui Yan, Dongyan Zhao, and Weinan E. Joint learning of response ranking and next utterance suggestion in human-computer conversation system. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 685–694. ACM, 2017.
- [197] Liu Yang, Hamed Zamani, Yongfeng Zhang, Jiafeng Guo, and W. Bruce Croft. Neural matching models for question retrieval and next question prediction in conversation. *arXiv:1707.05409*, 2017.
- [198] Peilin Yang and Hui Fang. University of delaware at TREC 2015: Combining opinion profile modeling with complex context filtering for contextual suggestion. In *Proceedings of the Text REtrieval Conference (TREC)*. NIST, 2015.
- [199] Peilin Yang, Hongning Wang, Hui Fang, and Deng Cai. Opinions matter: a general approach to user profile modeling for contextual suggestion. *Inf. Retr. Journal*, 18(6):586–610, 2015.
- [200] Zijun Yao, Yanjie Fu, Bin Liu, Yanchi Liu, and Hui Xiong. POI recommendation: A temporal matching between POI popularity and user regularity. In *Proceedings of the 16th IEEE International Conference on Data Mining, Barcelona, Spain*, pages 549–558, 2016.
- [201] Mao Ye, Peifeng Yin, Wang-Chien Lee, and Dik Lun Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 325–334. ACM, 2011.

- [202] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Zi Huang. A temporal context-aware model for user behavior modeling in social media systems. In *Proceedings of the International Conference on Management of Data (SIGMOD)*, pages 1543–1554, 2014.
- [203] Hongzhi Yin, Bin Cui, Yizhou Sun, Zhiting Hu, and Ling Chen. LCARS: A spatial item recommender system. *ACM Trans. Inf. Syst.*, 32(3):11:1–11:37, 2014.
- [204] Hongzhi Yin, Bin Cui, Ling Chen, Zhiting Hu, and Xiaofang Zhou. Dynamic user modeling in social media systems. *ACM Trans. Inf. Syst.*, 33(3):10:1–10:44, 2015.
- [205] Hongzhi Yin, Bin Cui, Xiaofang Zhou, Weiqing Wang, Zi Huang, and Shazia W. Sadiq. Joint modeling of user check-in behaviors for real-time point-of-interest recommendation. *ACM Trans. Inf. Syst.*, 35(2):11:1–11:44, 2016.
- [206] Hongzhi Yin, Weiqing Wang, Hao Wang, Ling Chen, and Xiaofang Zhou. Spatial-aware hierarchical collaborative deep learning for POI recommendation. *IEEE Trans. Knowl. Data Eng.*, 29(11):2537–2551, 2017.
- [207] Fajie Yuan, Joemon M. Jose, Guibing Guo, Long Chen, Haitao Yu, and Rami Suleiman Alkhaldeh. Joint geo-spatial preference and pairwise ranking for point-of-interest recommendation. In *Proceedings of the 28th IEEE International Conference on Tools with Artificial Intelligence, San Jose, CA, USA*, pages 46–53, 2016.
- [208] Quan Yuan, Gao Cong, Zongyang Ma, Aixin Sun, and Nadia Magnenat-Thalmann. Time-aware point-of-interest recommendation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 363–372. ACM, 2013.
- [209] Quan Yuan, Gao Cong, and Aixin Sun. Graph-based point-of-interest recommendation with geographical and temporal influences. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 659–668. ACM, 2014.
- [210] Hamed Zamani and W. Bruce Croft. Estimating embedding vectors for queries. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (ICTIR)*, pages 123–132. ACM, 2016.

- [211] Hamed Zamani, Michael Bendersky, Xuanhui Wang, and Mingyang Zhang. Situational context for ranking in personal search. In *Proceedings of the International Conference on World Wide Web (WWW)*, pages 1531–1540, 2017.
- [212] Hamed Zamani, W. Bruce Croft, and J. Shane Culpepper. Neural query performance prediction using weak supervision from multiple signals. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 105–114. ACM, 2018.
- [213] Hamed Zamani, Bhaskar Mitra, Xia Song, Nick Craswell, and Saurabh Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*. ACM, 2018.
- [214] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. *SIGIR Forum*, 51(2):268–276, 2017. ISSN 0163-5840.
- [215] Chenyi Zhang, Hongwei Liang, and Ke Wang. Trip recommendation meets real-world constraints: POI availability, diversity, and traveling time uncertainty. *ACM Trans. Inf. Syst.*, 35(1):5:1–5:28, 2016.
- [216] Jia-Dong Zhang and Chi-Yin Chow. Geosoca: Exploiting geographical, social and categorical correlations for point-of-interest recommendations. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 443–452. ACM, 2015.
- [217] Weishi Zhang, Guiguang Ding, Li Chen, Chunping Li, and Chengbo Zhang. Generating virtual ratings from chinese reviews to augment online recommendations. *ACM Trans. Intell. Syst. Technol.*, 4(1):9:1–9:17, 2013.
- [218] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the International Conference on Information and Knowledge Management (CIKM)*, pages 177–186. ACM, 2018.
- [219] Shenglin Zhao, Tong Zhao, Irwin King, and Michael R. Lyu. Geo-teaser: Geo-temporal sequential embedding rank for point-of-interest recommendation. In *Proceedings of the 26th International Conference on World Wide Web Companion, Perth, Australia*, pages 153–162, 2017.