

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Recommendation in Dialogue Systems

Permalink

<https://escholarship.org/uc/item/4rs1s3ms>

Author

Sun, Yueming

Publication Date

2019

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

RECOMMENDATION IN DIALOGUE SYSTEMS

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

TECHNOLOGY AND INFORMATION MANAGEMENT

by

Yueming Sun

September 2019

The Dissertation of Yueming Sun
is approved:

Professor Yi Zhang, Chair

Professor Lise Getoor

Professor Zhou Yu

Dilek Z. Hakkani-Tür, Ph.D.

Quentin Williams
Acting Vice Provost and Dean of Graduate Studies

Copyright © by

Yueming Sun

2019

Table of Contents

List of Figures	vi
List of Tables	viii
Abstract	xi
Dedication	xiii
Acknowledgments	xiv
1 Introduction	1
1.1 The Current Recommender Systems, Dialogue Systems, and Their Limitations	1
1.2 Benefits and Goals of Conversational Recommender Systems . . .	5
1.3 Contribution of the Dissertation	8
1.4 Outline of the Dissertation	9
2 Related Work	11
2.1 Recommender Systems	11
2.1.1 Content-based Recommender Systems	11
2.1.2 Collaborative Filtering Recommender Systems	13
2.1.3 Hybrid Recommender Systems	14
2.1.4 Deep Learning based Recommender Systems	15
2.2 Reinforcement Learning	16
2.2.1 The Markov Decision Process	16
2.2.2 Function Approximation and Deep Reinforcement Learning	18
2.2.3 Reinforcement Learning based Recommender Systems . . .	20
2.3 Dialogue Systems	21
2.3.1 The Chit-Chat, Information-Seeking and Task-Oriented Dialogue Systems	21
2.3.2 Components of Task-Oriented Dialogue Systems and End-to-end Task-Oriented Dialogue Systems	23

2.3.3	Conversational Recommender Systems	25
3	Conversational Recommendation Dialogue Data Collection	28
3.1	Overview	28
3.2	Related Work	29
3.3	Faceted based Recommendation Dataset Processing	31
3.4	Crowd Sourcing based Conversational Dialogue Dataset Collection	33
3.5	Summary	39
4	The Framework of Conversational Recommender System	40
4.1	Overview	40
4.2	The Conversational Recommender System Framework	41
4.2.1	Belief Tracker	41
4.2.2	Recommender Module	44
4.2.3	Deep Policy Network based Dialogue Manager	46
4.3	Experimental Setup	48
4.3.1	User Simulation	48
4.3.2	Baselines and Evaluation Methodology	51
4.3.3	Model Training Details	53
4.4	Experiment Results	54
4.4.1	Offline Experiments	54
4.4.2	Online User Study	60
4.5	Summary and Contribution	62
5	Personalization in Conversational Recommender System	65
5.1	Overview	65
5.2	Methodologies	67
5.2.1	Belief Tracker	67
5.2.2	Personalized Probabilistic Recommender	67
5.2.3	The Reinforcement Learning based Dialogue Agent	71
5.3	Experimental Settings	75
5.3.1	User Simulation	75
5.3.2	Evaluation Measures and Baseline for the Personalized Probability Recommender	75
5.3.3	Evaluation Measures and Baselines for the Dialogue Agent	76
5.4	Experiments Results	77
5.4.1	The Personalized Probabilistic Recommendation	77
5.4.2	The Personalized Conversational Recommender System	79
5.4.3	User Study	85
5.5	Summary and Contribution	87

6	Conclusion and Future Work	88
6.1	Contribution	88
6.2	The Limitations and Future Work	89
6.2.1	Limitation of the Collected Conversational Recommendation Dataset	90
6.2.2	Joint Learning the Recommender and the Dialogue Policy	91
6.2.3	Enabling More User Actions and Machine Actions	91
6.2.4	Integrating More Dialogue Functionalities	92
	Bibliography	94

List of Figures

3.1	The histogram diagram of the number of facet values of items on the MovieLens1M (upper) and Yelp Challenge dataset (lower). . .	32
3.2	The dialogue collection graphical user interface. On the left panel shows instructions and several examples. On the right panel is the dialogue window, where an MTurk is requested to paraphrase the schema into natural language user utterance.	35
4.1	The conversational recommender system overview	41
4.2	The structure of the proposed conversational recommender framework. The bottom part is the belief tracker, the top left part is the recommendation model, and the top right part is the deep policy network.	42
4.3	Comparisons on the three measures with different belief tracker accuracy.	59
5.1	The proposed framework. The bottom part is the belief tracker, the middle part is the personalized probabilistic recommender, and the top part is the dialogue manager.	66
5.2	Comparing the percentage of dialogue turns, the success rate and the average return of the PA and NPA at different number of dialogue turns, on the MovieLens1M dataset (the upper row) and the Yelp dataset (the lower row).	82

5.3 The Average Return obtained by using the Policy Gradient method and the A3C method. The x-axis is the total number of evaluation steps. The y-axis shows the Average Return evaluated on the whole dev dataset. 84

List of Tables

3.1	Basic statistics of the pre-processed MovieLens1M and Yelp recommendation datasets.	32
3.2	An example of a simulated dialogue schema for a user item pair in the MovieLens1M dataset	34
3.3	The crowdsourced dialogue scripts using the schema in Table 3.2	36
3.4	Statistics of the two conversational recommendation datasets, comparing to the DSTC2 dataset.	38
3.5	Example dialogues simulated using the method discussed in this chapter. The left dialogue is a MovieLens1M example, the right dialogue is a Yelp example.	38
4.1	Comparisons of CRM and the baselines on Average Return, Average Turn, Success Rate, Wrong quit rate, and Low rank rate. Results in bold are significantly better than baselines at 0.01 level.	55
4.2	The dialogue with CRM agent (left) is more aggressive than the one with the MaxEnt agent (right).	56
4.3	The MaxEnt agent (right) fails early due to the belief tracker error, but the CRM agent (left) is able to by pass the failure case and succeed.	57
4.4	The MaxEnt agent (right) fails due to the belief tracker error, but the CRM agent (left) asks a different facet at the second turn, avoids the belief tracker misunderstanding, and succeed with shorter turns.	57

4.5	Comparisons of CRM and the baseline with different user behaviors of checking the recommendation results, on R (Average Return), T (Average Turn), and S (Success Rate).	58
4.6	Results of R (Average Return), T (Average Turn), and S (Success Rate) with different maximum reward C	60
4.7	Model performances of three measures with different cutoff threshold K	60
4.8	Comparison of CRM and the MaxEnt method on R (Average Return), T (Average Turn), and S (Success Rate) in online user study.	62
4.9	User study example dialogues of CRM and the MaxEnt methods. The facet values extracted by the belief tracker module are shown in the parentheses.	63
5.1	Comparison of the Non-personalized Recommender (NPR) and the Personalized Recommender (PR) with baselines. Evaluated on the MRR@100 of the target item, when different number of facets are known. The top part is on the MovieLens1M dataset, and the bottom part is on the Yelp dataset. Results in bold are significantly better than baselines at 0.01 level.	78
5.2	The conversational recommendation dialogue performance of the Non Personalized Agent (NPA) and the Personalized Agent (PA) with the simulated user on the MovieLens1M Dialogue Dataset (the upper part) and on the Yelp Dialogue Dataset (the lower part). Results in bold are significantly better than baselines at 0.01 level.	80
5.3	Comparison of example dialogues between PA and the NPA on the MovieLens1M dataset (the top row) and the Yelp dataset (the bottom row).	81
5.4	Comparing the conversational recommendation performance of the PA and NPA interacting with simulated users having different continue probability p_c	83

5.5	The comparison of NPA and PA in online user study on the MovieLens1M dataset.	85
5.6	The comparison of NPA and PA in online user study on the Yelp dataset.	85
5.7	User study example dialogues of PA and NPA on the MovieLens1M dataset (the top row) and the Yelp dataset (the bottom row). . .	86

Abstract

Recommendation in Dialogue Systems

by

Yueming Sun

Dialogue system has been an active research field for decades and is developing fast in recent years, due to the recent breakthrough of the deep learning techniques. How to make recommendations in dialogue systems is attracting increasing attention because such systems could meet various user information needs and have much commercial potential.

Current dialogue system researches typically focus on building systems for social conversation, question answering, and performing specific tasks. However, making recommendations to users, as important information need, has not been intensively researched. Meanwhile, traditional recommender systems are usually developed for non-conversation scenarios. In this dissertation, we explore how to integrate these two systems into one framework that specifically aims at making recommendations in dialogues. Such a system helps users find items by chatting with users to understand their preferences and recommending accordingly.

First, we build conversational recommendation datasets, because existing dialogue datasets do not have user-item preference information or the dialogue utterances discussing facets of items, and current recommendation datasets do not have dialogue scripts associated with each user-item pair. We build the datasets by requesting crowdsourcing workers to compose dialogue utterances based on schemas and then use the delexicalization approach to simulate dialogues with the collected utterances. The datasets are used to train the natural language understanding component and provide recommendation information for our system.

Based on collected datasets, we propose a reinforcement learning based conversational recommendation framework. Such a framework has three components, a belief tracker, a dialogue manager, and a recommender. The dialogue agent learns to first chat with a user to understand her preferences, and when it feels confident enough, it recommends a list of items to the user. We conduct both offline and online experiments to demonstrate the effectiveness of the framework.

We further extend this framework with a personalized probabilistic recommender module. This recommender learns to predict the probability of a user likes an item given the dialogue utterance information and the personalized user preference information. By leveraging this hybrid information, the recommendation and dialogue performances are further improved. We evaluate the dialogue agent’s strength in various simulated environments as well as in online user studies and demonstrate the advantages of this approach.

To Xingxing & my mother

Acknowledgments

I would like to express my sincerest gratitude to my advisor Prof. Yi Zhang for her patient guidance and continuous support in my Ph.D. research. Her inspiring vision and immense knowledge guided me to conquer research problems, and her encouragement helped me keep moving forward. Without her persistent help, I can never finish this whole journey.

Besides my advisor, I would like to thank my committee members, Prof. Lise Getoor, Prof. Zhou Yu, and Dr. Dilek Z. Hakkani-Tür, for their reviewing of this dissertation, the discussions we have, and their insightful suggestions for the improvement of this work.

I want to thank members of Information Retrieval and Knowledge Management (IRKM) lab, Qi Zhao, Yunfei Chen, Jian Wang, Lanbo Zhang, Xilian Li; also my friends Tian Jin, Jie Li, Xing Yi, and Huajun Zeng. Thank you for discussing research questions with me, and for the happy time we spent together.

I am very grateful to the support of my family. My mother always believes in me and encourage me to do what I want. The love she gives me is unconditional. Most of all, I want to express my deepest gratitude to my wife Xingxing. She supports me with all her heart. The company she gives me in my hard time is invaluable. This dissertation is dedicated to my family.

Chapter 1

Introduction

1.1 The Current Recommender Systems, Dialogue Systems, and Their Limitations

The modern world has witnessed the prosperity of information technologies. Recommender Systems play an increasingly important role in finding users their preferred items from the huge amount of information. These systems find items that match the personalized preferences of users efficiently and precisely. Recommender systems are widely used in various applications, for instance, music streaming services [10], online movie rating databases [127] [24], restaurants recommendation websites [25], online shopping websites [26], news online portals [52], and video streaming services [14]. These systems use different types of information for the recommendation purpose, such as the past items one has rated, the URLs one has clicked, the orders one has placed, and the comments one has scripted.

Recommender systems are mainly divided into three categories [132] [1]: the content-based system, the collaborative filtering system, and the hybrid system.

The content-based systems recommend items that are similar to those ones liked by users before. The collaborative filtering systems generate recommendations based on the preferences of users who have similar tastes with the target user. The hybrid systems integrate multiple systems in systematic ways and usually have the strengths of each individual system.

The Dialogue System is one of the important applications of artificial intelligence. It has been researched and developed for decades. In such a system, an end-user and a dialogue agent take turns to communicate in natural language, so that the system can fulfill the user's information need. Generally, dialogue systems can be categorized into three types [21]: the open domain dialogue system (the chit-chat system) [114] [45], the question answering system (the information-seeking system) [77] [79], and the task-oriented system [134] [18]. With the recent breakthrough of the deep learning technology and the significant increase of computing power, researches of dialogue systems have entered a new era. The collaboration of the research and industry has successfully spawned many popular conversational AI, for example, the Alexa of Amazon, the Siri of Apple, the Google Assistant, the Facebook Messenger and the Cortana of Microsoft. These chatbots are implemented on different platforms, such as mobiles, home devices, and webpages.

Dialogue systems are becoming indispensable tools in our life. First, the dialogue system fulfills a wide range of user information need [21], for example, a user can search news, ask the weather, track a package status, book a ticket, and place an order. Second, the dialogue system is an important entrance of online services and can bring the business provider a lot of traffic. Once a user initiates an interaction with a dialogue agent, the agent can proactively provide various follow-up services, such as making recommendations, to keep the user in-the-loop,

which can create many business opportunities. Third, dialogue systems such as the virtual customer service agents can provide services 24/7 and can handle conversations at scale, which are suitable to operate repetitive works and save cost for business owners. Moreover, the dialogue systems can free users hand in a variety of scenarios, e.g., find a route while a user is driving, control the home devices while a user is arriving home.

Techniques such as reinforcement learning and the sequence-to-sequence [107] models are widely used in the dialogue system researches. Reinforcement learning views the dialogue procedure as a sequence of decision makings. At each turn, it maintains a description of the dialogue state, based on which it samples an action, according to a learning policy, so that the action is deemed to be the most long-term beneficial [135]. Sequence-to-sequence models view the conversation as an encoding-decoding process, for example, the user utterance is modeled as the input sequence, based on which the model generates an output sequence, which is a natural language response. These models maintain some sorts of memory, for instance, the dialogue history, which is updated each turn. Much information could be encoded in this way, depending on the details of modeling approach [119]. These models are usually learned by optimizing some object functions, which are specifically designed for the problems being studied.

However, not much work has been done towards integrating the recommender system and the dialogue system to build the conversational recommender systems, which are very important to contemporary life and have big potential. According to the statistics¹, for example, in 2018, there are around 16 million salespeople in the US. As virtual sales agents, conversational recommender systems can provide services to users anytime, anywhere. They can be complementary to the salespeople, or to some extent, be their substitutes. There are huge business opportunities

¹<https://www.bls.gov>

in this field. The conversational recommendation information need has not been properly handled by existing recommender systems or dialogue systems.

On the one hand, the recommender systems research community did not address the distinctive characteristics of the dialogue scenario. In the dialogue scenario, the user feedback is usually given in real-time, while traditional recommender systems use static or historical information a lot, i.e., reviews. User feedback is relatively clear in dialogue systems; for example, a user will try to describe her need in explicit utterances. However, the traditional systems deal with less explicit information, for instance, ratings and reviews [38] [59], which contain much noise, or user clicks and impressions, which are implicit feedback. Moreover, a user can provide the information required by the dialogue systems, especially the information that is deemed as crucial for the system to make better services. There is one line of research in traditional system literature denotes the conversational recommendation as critique-based multiple round recommendations [7] [85]. However, there are no explicit natural language dialogues in such kinds of systems.

On the other hand, the dialogue system research community did not address the recommendation information need too much. Each of the three types of dialogue systems has different research focuses, and none of them treats making recommendations in the dialogue as a major task. The chit-chat systems focus on generating meaningful [114], diverse [43], and persona consistent [44] responses. The question answering systems focus on finding correct answers to specific questions and are usually evaluated by measures such as precision and recalls. They tend to focus on the natural language processing aspects of the dialogue system. For example, generate answers for factoid questions using a knowledge base, retrieve answers from a set of candidates, and generating answers from a segment

of the given text, which is also known as reading comprehension.

Task-oriented dialogue systems focus on completing specific tasks for users, such as booking a movie ticket or reserving a table in a restaurant. Though recommendation can be viewed as a particular type of task, there are no previous works that are built with making personalized recommendations as the direct goal. Many works format the tasks as slot-filling procedures, and dialogue agents collect the slot values while conversing with users. When the required information is collected, a knowledge base is looked up to find the target items for users [134]. The number of instances in the knowledge base is usually much smaller than that of a traditional recommender system [119]. Decision makings in such systems are not recommendation oriented [19] [133]. Moreover, the generated results are not personalized [18]. Different users get the same set of results, and therefore, these systems are information access systems rather than personalized recommender systems.

Based on the discussions above, how to make recommendations in a dialogue system remains an open question to both the recommender systems and the dialogue system research communities.

1.2 Benefits and Goals of Conversational Recommender Systems

In this dissertation, we explore how to build dialogue systems that aim at making personalized recommendations for users. Our systems are studied and built based on the combination of recommendation research and dialogue system research. We argue that conversational recommender systems have the following benefits and characteristics:

First, conversational recommender systems can bring potential business values to both end-users and service providers. For service providers, a successful recommendation can lead to a conversion to business success naturally. There are a variety of scenarios; for example, a good product recommendation can lead to purchasing; a successful movie recommendation can trigger a ticket booking; a restaurant recommendation can be converted to a reservation. High-quality recommendations can increase customer satisfaction and loyalty to the service. For end-users, conversational recommender systems are handy tools which can help them find target items via close cooperation. On the traditional website, a user is all on her own, for example, navigating through links on web pages to find the ideal product. However, a conversational recommendation agent can guide the user through such a process, collect necessary information, and make personalized suggestions. Users can express their needs in natural languages. Besides, users do not always precisely know which items they want; they might only have some preferences on several attributes. Just like a real salesperson, a conversational recommender is an expert in this domain and can assist users to elicit and clarify their needs gradually.

Second, the dialogue system and the recommender system bring merits to each other. For the recommendation, the dialogue system can provide real-time user feedback. Traditional recommender systems often use users' historical interaction data, while dialogue scripts provide the real-time user need and are more explicit. Besides, the current user interest may not always be consistent with the user's past preferences, and therefore, the current and historical user interest is complementary information to each other, and together lead to better recommendation results. For the dialogue system, a good recommender can bring more diversities in the conversation, such as increasing the length of interactions with users, sug-

gesting more types of services for users, and attracting users to be more engaged with the conversation. Current dialogue systems are relatively passive in completing specific tasks originated by users. Recommender systems allow dialogue agents to have more initiative and behave proactively to create more business opportunities in assisting users.

Third, the conversational recommender system should learn when to make a recommendation. In the conversation, the dialogue agent should first learn to talk with users about the features of the items, in order to collect sufficient user interests, and then make the recommendation when it is confident enough. If the recommendation is made too fast, the target item might not be ranked high enough, and the recommendation can fail. However, if the dialogue agent asks too many questions, users may get bored and leave the dialogue before the recommendation could be made. Moreover, the dialogue system can not always understand users correctly. So the longer the conversation is, the higher the chances of making mistakes. Therefore it is a trade-off between collecting enough information and making a good recommendation quick enough.

Fourth, conversation recommender systems should support rich interactions with users. For example, request and inform a facet value, negate a previous constraint, and confirm questions. Users should be able to modify their preferences at any turn. There could be multi-round recommendations in a single dialogue session. Moreover, dialogue agents should be able to answer different types of user questions, for example, general questions before the selection of an item or specific questions related to a selected item. The system should also analyze user logs to build user profiles to provide better-personalized services. For example, some users might be more patient and talkative, while others might be succinct and less patient. The above features can be better achieved via running the

system against the real users to collect rich user interactive information. Due to the limitation of our datasets, we leave these further extensions to future work.

1.3 Contribution of the Dissertation

We build a conversational recommender system to make successful recommendations for users. In our system, the dialogue agent first chats with users to understand their preferences gradually and then make recommendations accordingly.

The contributions of this dissertation are three-folds. First, we build conversational recommendation datasets based on two traditional recommendation datasets, with crowdsourcing techniques. On the one hand, existing recommendation datasets are in the non-dialogue problem domain and do not have the dialogues associated with them. On the other hand, current dialogue datasets do not have personalization information; for instance, they do not contain the user-item pair information. In order to conduct the experiments in this dissertation and to facilitate researches in this direction, we design online experiments to request the crowdsourcing workers on the Amazon Mechanical Turk platform to compose dialogue scripts. In this way, we build the conversational recommendation datasets in two domains. We use these datasets to train modules for understanding user requirements in dialogue utterances, as well as making recommendations in the dialogue.

Second, we propose a framework that is specifically tailored for making recommendations in the dialogue system. This framework combines the strengths of dialogue systems and recommender systems to build a dialogue agent to chat with users in natural language in order to find the target items for them. The framework has three components: 1) a belief tracker, for taking the user utterance

to compute a dialogue state; 2) a dialogue manager that decides what actions to take given the dialogue state; 3) a recommender module to recommend a personalized item list using the dialogue states. The dialogue manager is a reinforcement learning based component. Under such a framework, our system learns to first chat with a user to collect the facet value constraints on the items this user wants. When the system is confident enough, it will make a recommendation to the user. We simulate users for the dialogue agent to interact with to boost the training of the model. We then perform both offline and online experiments to demonstrate the effectiveness of our proposed method.

Third, we propose a personalized probabilistic recommendation module for the conversational framework. This model integrates the personalized information, which is the historical user preferences, and the real-time user constraints enclosed in the user utterances to compute the conditional probability of all the items given a user and a dialogue. We compare this model to the state-of-the-art recommendation models to demonstrate the advantages of our approach. We then integrate the model into the conversational recommendation framework and perform different experiments to demonstrate that it further improves the recommendation and dialogue performance. We also conduct user studies to testify the effectiveness of our model with real users.

1.4 Outline of the Dissertation

The rest of the dissertation is organized as follows: Chapter 2 discusses the related work, including the recommender systems, the reinforcement learning method, the dialogue system, and their intersections. Chapter 3 introduces the conversational recommendation dataset collection with the help of the crowdsourcing technique. Chapter 4 describes our proposed conversational recommendation

framework and its implementation. Chapter 5 describes how to address the personalization in the proposed framework better to reach improved recommendation and dialogue performance. Chapter 6 summarizes major contributions and discusses our vision of future improvement directions.

Parts of the dissertation have been published in conferences. Chapter 3 is based on a paper published in the 10th ACM Conference on Recommender Systems in 2016 [106] and a paper published in the 41st International ACM SIGIR Conference in 2018 [105]. Chapter 4 is based on the paper [105]. Chapter 5 is based on the most recent work that has yet to be submitted for publication.

Chapter 2

Related Work

2.1 Recommender Systems

Recommender systems are important information tools to filter and find items, usually from a large number of candidate items, that are of interest to different users [53]. They are ubiquitous on the Internet and have become indispensable to our daily life. The research in recommender systems dates back to the mid-1990s [83] and is still one of the most rigorous research areas today. Recommender systems are generally categorized into three categories: the content-based recommender systems, the collaborative filtering recommender systems, and the hybrid recommender systems [132] [1].

2.1.1 Content-based Recommender Systems

Content-based recommender systems find items for users according to similar items that are liked by them before [1] [75] [53]. One typical example is the Vector Space Model-based recommendation, in which users and items are represented as vectors of words, and the user-item preference is modeled by the closeness of

the representation vectors on certain distance measures, for instance, the cosine similarity. Each element in the vector corresponds to a word in the vocabulary of corpus and is represented by the Term Frequency (TF) - Inverse Document Frequency (IDF) weight [1] [53], which are defined as:

$$TF_{i,w} = f_{i,w} \quad (2.1)$$

$$IDF_w = \log\left(\frac{N_D}{D_w}\right) \quad (2.2)$$

$$w_{TF-IDF} = TF \times IDF = f_{i,w} \log\left(\frac{N_D}{D_w}\right) \quad (2.3)$$

where $f_{i,w}$ is the raw count of a word w in a document i (or could be a normalized quantity). D_w is the document frequency of the word w , and N_D is the total number of documents in the corpus.

Another example is the machine learning based Naive Bayes approach [74] [69]. Again, an item i is represented as a set of words, and the problem is to compute the probability of the item belong to a class c (liked or not liked). This is modeled by the conditional probability $p(c|i)$, according to the Bayesian Theorem:

$$p(c|i) = \frac{p(i|c)p(c)}{p(i)} \quad (2.4)$$

where $p(c)$ is the prior probability of class c , and $p(i|c)$ is the conditional probability of observing an item given the class c . This approach has a strong assumption that the $p(i|c) = \prod_{w_k \in V^i} p(w_k|c)$, where w denotes words in the vocabulary V^i of the item i . Therefore, both the $p(c)$ and $p(i|c)$ could be estimated from the dataset. This approach shows good performance in real applications [74].

Content-based recommender systems have several limitations [1]: (1) the performance of the recommendation is usually determined by the strength of content

features used [97]; (2) lacking novelty, as items dissimilar to a user’s historical preferences would be hard to be recommended [4]; (3) the cold start problem, can not make proper recommendations for a new user whose historical information is insufficient [39].

2.1.2 Collaborative Filtering Recommender Systems

Collaborative filtering recommender systems assume that users have similar preferences on some items would also have similar tastes on some other items [86]. Generally, they can be divided into the memory-based and model-based approaches [104]. Take the user-based collaborative filtering method as an example of the memory-based approaches [86] (the item-based method can be derived similarly). To predict a rating of a user u to a target item i , this approach first finds a subset of users V who have rated this item and have similar tastes with the target user. It then computes the similarity between the target user and each user v in V . Commonly used similarity scores are the cosine similarity and the Pearson correlation coefficient [17], where:

$$\text{cosine}(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} r_{vi}}{\sqrt{\sum_{i \in I_u} r_{ui}^2} \sqrt{\sum_{i \in I_v} r_{vi}^2}} \quad (2.5)$$

$$\text{Pearson}(u, v) = \frac{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{ui} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{vi} - \bar{r}_v)^2}} \quad (2.6)$$

and the predicted rating of the target item by the target user is the weighted sum of ratings of the target item given by users in the neighboring set V .

$$r_{ui} = \frac{\sum_{v \in V} r_{vi} \text{score}(u, v)}{\sum_{v \in V} |\text{score}(u, v)|} \quad (2.7)$$

The model-based collaborative methods predict ratings using a learned model.

One of the classic models is the Matrix Factorization model [38], which became widely recognized due to the Netflix Prize competition¹. The main idea is to map users and items into a low dimensional latent space, with the latent factors encoding the users and items interactions. Specifically, this mechanism represents the rating of a user to an item to be the dot product of the K dimensional latent vectors of the user and the item. Many popular models are developed based on this, such as the SVD++ [37], the ListRank-MF [99] model, the BPR model [82], and the Factorization Machine [81]. Collaborative filtering recommender systems also have a few limitations: (1) the cold start problems, making predictions for new users and new items when not enough ratings are available; (2) the data sparsity problem, because enough ratings have to be observed to compute the user preferences. In this dissertation, we propose the personalized probabilistic recommendation model by extending the Matrix Factorization model. We use the Factorization Machine, the BPR model, and the ListRank-MF as our baselines.

2.1.3 Hybrid Recommender Systems

The hybrid recommender systems combine the strength of several systems, e.g., the content-based recommender systems and the collaborative filtering recommender systems into a unified model which has the mixing power of each approach [8]. The Fab system [3] is one of the earliest hybrid systems. De Campos et al. [16] presented a Bayesian Hybrid recommendation model using a probability reasoning technique to compute the conditional probability of a rating, using the evidence from the network topology of the content-based and the collaborative based recommendation components. Our proposed model is a hybrid approach as we leverage the collaborative latent factors of the users and items, as well as the

¹<https://www.netflixprize.com/>

content features contained in the dialogue utterances into our model. This brings us benefits of recommending items based on the real-time user needs as well as the personalized past user preferences. More details will be introduced in Chapter 5.

2.1.4 Deep Learning based Recommender Systems

Due to the recent success of deep learning techniques, the deep neural networks have been successfully applied to many traditional research areas [40], including recommender systems. Traditional machine learning methods normally depends on the carefully designed features, while deep learning methods can automatically learn representations from the raw data. Convolution Neural Network (CNN) [41] and Recurrent Neural network (RNN) [33] are two of the widely used networks, which are good at capturing spatial representations and temporal features, respectively. There are a lot of works in applying the deep neural networks for the recommender systems [27] [12] [2] [23]. In this dissertation, we mainly use Multi-layer Perceptron (MLP) and RNN structures [33]. MLP is a feedforward neural network which has one or more hidden layers, with non-linear activation functions for each layer. When properly designed, MLP can be used to learn an arbitrarily complex function. RNN is a cell structure with the inside loops, which serve as the internal memory structures. When expanded across the time steps, RNN is transferred into a feedforward neural network across the temporal steps. RNN is widely used to model the problems with sequential information, for example, the conversational dialogue. Readers may refer to [132] for a much more detailed review.

2.2 Reinforcement Learning

Reinforcement Learning is one type of machine learning approaches, peered with supervised learning, and unsupervised learning methods [109]. This learning approach is suitable for problems which require sequential decision makings. Usually, such problems are interpreted as an agent, or the learned model, constantly interacting with the environment. In this process, the agent gets step-wise rewards or episodic rewards, which are used to update the agent's decision policies. The agent's goal is to maximize the expected total rewards. Therefore the rewards are usually designed to include the learning target of research problems being studied [72] [135] [134]. There are two major advantages of reinforcement learning [109]. First, reinforcement learning focuses on long term learning. It not only looks at the immediate reward but also all the subsequent rewards or the delayed reward. Second, reinforcement learning is a trial-and-error learning process which balances the exploration and exploitation.

2.2.1 The Markov Decision Process

Reinforcement Learning is usually formalized as Markov Decision Processes (MDPs) [126]. An MDP is described by a tuple of $(s_t, a_t, r_t, \pi, \gamma)$. At time step t , the state s_t is a description of the environment. The action a_t is selected by the agent, according to the learned policy π , which is a conditional probability $\pi(a_t|s_t)$. After taking an action, the agent receives a reward r_t from the environment. And the state s_t is updated to a new state s_{t+1} at time step $t + 1$, affected by the action taken by the agent. Meanwhile, the agent uses the delayed reward signals to update its policy π so that in the long run, it can get the maximum expected total rewards. Note that the agent learns through interacting with the environment, but it can not change the environment settings. The MDP contains a trajectory of

$s_t, a_t, r_t, s_{t+1}, a_{t+1}, r_{t+1}, \dots$. The learning object, which is the expected discounted return, is defined by:

$$G_t = r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} + \dots \quad (2.8)$$

where γ is a discount factor and is in the range of $(0, 1)$. If γ is close to 0, the agent focuses more on the near future, and if it is close to 1, the agent focuses more on the long term benefits.

In order to find a good policy, the reinforcement learning agent learns to estimate value functions, including the state-value function $v_\pi(s_t)$ and the action-value function $q_\pi(s_t, a_t)$, which measures how good it is to be in a given state or a state-action pair, respectively. According to the assumption of Markov property, the value functions only depend on the immediately preceding state and action, not on the other previous states and actions [109]. Thus the Bellman equations hold, with the state-value function:

$$v_\pi(s_t) = E_\pi(G_t | s_t) = \sum_{a_t} \pi(a_t | s_t) \sum_{s_{t+1}, r_t} p(s_{t+1}, r_t | s_t, a_t) [r_t + \gamma v_\pi(s_{t+1})] \quad (2.9)$$

and the action-value function, also known as Q-function:

$$q_\pi(s_t, a_t) = \sum_{s_{t+1}} p(s_{t+1} | s_t, a_t) r_t + \gamma \sum_{s_{t+1}} p(s_{t+1} | s_t, a_t) \sum_{a_{t+1}} \pi(a_{t+1} | s_{t+1}) q_\pi(s_{t+1}, a_{t+1}) \quad (2.10)$$

Solving the reinforcement learning problem is equivalent to finding the optimal value functions:

$$\begin{aligned} v_* &= \max_\pi v_\pi(s_t) \\ q_* &= \max_\pi q_\pi(s_t, a_t) \end{aligned} \quad (2.11)$$

Methods commonly used for solving the reinforcement learning problems can be divided into the Monte Carlo methods, which learn from the episodic experiences, and the Temporal Difference (TD) methods, which learn partially from

the *estimates* of the raw experiences. Specifically, the off-policy TD method, also known as the Q-learning, is a significant breakthrough of the reinforcement learning [116]. As a model-free method, the Q-learning method learns without waiting till the end of the episode and does not follow the policy in the updating of the learning process, and therefore has few constraints in the learning process. It has the updating form:

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[R_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (2.12)$$

Note that the choice of the action a at state s_{t+1} does not follow the policy being learned, so it is an off-policy method, which is a crucial simplification of the learning process [109] [116].

2.2.2 Function Approximation and Deep Reinforcement Learning

Function approximation further enhances the generalization ability of reinforcement learning and does not rely on the strong assumptions of Markov property. In reality, the problems we tackling usually have a large (discrete or continuous) state space, so that the exact modeling of the value function is infeasible. Moreover, the training data is usually very limited comparing to the complexity of the problem domain, and can not generalize well enough to the unseen states [109]. By borrowing the ideas from supervised learning, the function approximation allows a model to use parameterized functions to represent the value functions and can obtain good generalization ability. Due to the recent breakthrough in deep learning, deep neural networks are widely used as function approximators, which leads to the prosperity of the Deep Reinforcement Learning technique. One of

the remarkable success of the deep reinforcement learning is by Google’s DeepMind team, who has trained an agent to play the Game of Go and has gained the superhuman skills [68] [100] [102].

A series of works kept pushing forward the-state-of-the-art of Deep Reinforcement Learning. First, the deep Q-learning is proposed in [67] where the authors use a deep convolutional neural network as the function approximator of the Q function. They came up with the experience replay technique [50], which stores the experience in a buffer, and then samples instances from this buffer uniformly to train the Q network. This technique solves the data correlation problem, which often leads to the divergence of the learning process. Prioritized experience replay is then developed by [92] to improve experience replay. By introducing a prioritized sampling strategy, this approach samples the experiences that having a higher TD-error with higher priorities, which speeds up the learning and gets improved performance. The traditional Q-learning suffers from the overestimation problem [111], which is caused by the fact that the same Q function is used for both the evaluation of action values and the selection of actions. Double Q-learning [112] uses two networks to decouple the action selection and the action value evaluation, and therefore mitigate the overestimation problem. The Asynchronous Advantage Actor-Critic (A3C) approach [66] combines the advantages of the above models. This approach runs multi-threads on a single machine to train multiple agents at the same time, and update their learned parameter interchangeably. The updates are accumulated over several steps before applying to the global network. This mechanism solves the data correlation problem and the overestimation problem at the same time, leading to a model with superior performance. Our model in Chapter 5 adopts this approach as the reinforcement learning algorithm and has achieved better performance. Moreover, the Deep Deterministic Policy Gradient

(DDPG) approach [49] explored in the continuous action space scenario by using the deep neural networks as the function approximators. And [31] combines the above improvements to reach a more powerful state-of-the-art.

2.2.3 Reinforcement Learning based Recommender Systems

Researches on using reinforcement learning for recommender systems usually model recommendation as an N-item-gram problem: the recommendation process is viewed as recommending the next item(s) based on historical items, and the length of the past item sequence is usually limited to a fixed number N . By considering the recommendation as a sequential decision-making process, it is natural to model the historical item sequences as states, and the recommending of the next item or items as actions. However, the number of items is big, and the item combinations are enormous, which makes the problem having nearly infinite state and action space. Thus approximations and simplifications are used in the modeling to keep the problem manageable. Moreover, the rewards must be specifically designed to match the learning objects.

[96] is one of the notable early works of using reinforcement learning in the recommender systems. It models the recommendation process as an MDP process - recommend the next item(s) (n-item-gram) according to the previous k items. They applied the model on the real online bookstore website and showed the advantage of the reinforcement learning based recommender system over a traditional recommendation predictor. Taghipour et al. [110] also use the n-item-gram paradigm to recommend the next web page based on the previous k browsed pages. The reward is designed to encourage the recommended page being similar to the historical pages and increasing the time a user spent on the page.

More recently, deep neural networks are widely used as function approximators. Zheng et al. [136] proposed a deep reinforcement learning based news recommendation system. They use the double Q-learning framework, and the reward is designed to incorporate both the user clicks and the user activeness (user return time) to help improve the recommendation. Zhao et al. [135] developed a deep reinforcement learning based page-wise recommender system. They encode a two-dimensional page of items into the continuous embedding space and use both GRU and CNN network to capture temporal and spatial signals. By using the historical browsing/click data, they modeled the page-wise user behavior as an n-item-gram problem and used the DDPG to train a recommendation model.

2.3 Dialogue Systems

Dialogue systems have been an active research area for many decades. Such a system usually involves an end-user and a dialogue agent. By communicating via natural language interactively, the agent helps one to fulfill the information need.

2.3.1 The Chit-Chat, Information-Seeking and Task-Oriented Dialogue Systems

By the application scenarios, dialogue systems can be divided into the chit-chat dialogue system, the information-seeking dialogue system, and the task-oriented dialogue system.

The chit-chat dialogue systems aim at generating natural and appropriate responses for users. Vinyals et al. [114] built a chit-chat agent using a sequence-to-sequence model on the IT troubleshooting and the subtitle dataset. Li et al. [45] proposed the modeling of mutual information in the learning object [43] to

increase the diversity of the conversation. Zhang et al. [131] modeled the persona information for both user and agent in a dialogue, and show that the resulting chit-chat dialogues are more engaged with topics.

Information-seeking dialogue system or the question-answering systems respond to user questions with retrieved or generated answers using different kinds of data resources. [77] collect a question answering dataset with the Wikipedia articles, and the answers are sub-segments of the article. This work has triggered a significant amount of researches in the question-answering field. Reddy et al. [79] explore the conversational questions-answering domain, in which a sequence of follow-up questions are given based on a passage, and the answers are free-form text with evidence in the passage. [93] build a factoid question-answering corpus using facts from the Freebase.

Task-oriented dialogue systems are used to complete specific tasks for users, such as booking movie tickets. It has three major components [121] [126]: a Natural Language Understanding (NLU) module, which learns to parse the natural language sentences (or speeches) into meaningful representations, such as real vectors; a Dialogue Manager (DM) module, which takes the representation vector as input to compute a dialogue action; and a Natural Language Generation (NLG) module, which generates responses to the end-users, based on the action selected by the dialogue manager. There are component-wise approaches and end-to-end approaches in the building of the task-oriented dialogue systems.

2.3.2 Components of Task-Oriented Dialogue Systems and End-to-end Task-Oriented Dialogue Systems

Natural Language Understanding

The Natural Language Understanding (NLU) and Dialogue State Tracking (DST) are closely associated with each other [21]. This module is responsible for understanding the end-user input and generating a belief vector that can be interpreted by the dialogue manager. Mesnil et al. [63] modeled the NLU as a sequence labeling problem and different types of deep neural networks are used. [70] explores transferring knowledge in dialogue state tracking using delexicalised features. They train an RNN model on one domain and continue to train the model specifically on the other domain data. [71] combines the NLU and the DST models, and learn a joint Neural Belief Tracker to convert the user utterances into belief vectors. Moreover, the Dialogue State Tracking Challenge (DSTC) and its recent rebranded Dialog System Technology Challenges have been held for many years to research on a wide range of NLU and DST related problems².

Dialogue Manager

The dialogue manager is responsible for finding a dialogue policy that decides which action to take given the current dialogue state [123] [119]. Different methods have been used, for example, the agenda-based method [91] and the rule-based method [51] [124]. One of the widely used approaches is to learn a policy with the reinforcement learning framework, due to its aligning with the dialogue procedure [21]. [90] [121] are among the early works to model the dialogue system with the Partially Observed Markov Decision Process (POMDP), pointing out a promising direction of researches in this field. This framework has several advantages: (1)

²<https://www.microsoft.com/en-us/research/event/dialog-state-tracking-challenge/>

explicitly modeling the uncertainty of the user utterances, (2) stochastic modeling of the dialogue action given the dialogue state, and (3) being able to use the reward associated with the state action pairs [121] to guide the finding of a good dialogue policy. A comprehensive review of the dialogue manager is out of the scope of this thesis, and the readers may refer to [62] [126] [11] [21] for more details.

Natural Language Generation

The natural language generation component is used for taking the action selected by the dialogue manager and generating natural language responses to users. Traditional approaches break down the task into a three-step process: text planning, sentence planning, and realization [115]. Rule-based methods and template-based methods are used by early works [103]. Later statistical approaches and data-driven methods are used to decrease the handcrafting workload [73] [57] [87]. More recently, the neural network based methods mitigate the challenges of expensive handcrafting and can generate responses with better language variation and naturalness. Wen et al. [118] proposed a Semantically Conditioned LSTM (SC-LSTM) network to model the sentence planning and surface realization uniformly. They later extended this work to explore multi-domain NLG by first training the SC-LSTM parameters on the counterfeiting data and then to fine-tune the network on a small amount of domain-specific data [117].

End-to-End Dialogue System

Another important line of research manages to jointly train different components of dialogue systems in an end-to-end data-driven way. The end-to-end models have the advantage that the components of the system are differentiable and are optimized against the learning object directly with the gradient descent

techniques. There are many works in this direction. For example, Zhao et al. [134] proposed an end-to-end framework to jointly train the belief tracker and the RL based dialogue manager and demonstrate the superior of joint training on a 20 Question Game. [119] proposes a sequence-to-sequence model to train the dialogue system end-to-end, including modeling the internal belief states and intent distributions and decoding the DB query vectors into the dialogue responses. [18] jointly trains the NLU and DM modules in a reinforcement learning framework, to build a movie information access system, in which the users can find an item in the knowledge base.

2.3.3 Conversational Recommender Systems

There is one line of research of the conversational recommendation focusing on researching the critiquing-based systems [53]. More recent research, including this dissertation, focusing on how to make recommendations in natural language dialogue systems.

Critique-based conversational recommendation

In these systems, the conversation denotes making multiple cycles of recommendations for users. Interactions with users are mainly through graphical user interfaces presented on different platforms, e.g., on mobiles or webpages, rather than through natural language dialogues [7] [85]. Critiques are facet (attribute) value constrains of items. A user can modify the critiques, e.g., change the value or the scales, to browse the recommended results. One can continue to modify the critiques based on his preferences and the recommended items to update recommendations. This process continues until the user finds the target item in the results.

Research questions in this field include [53]: (1) how to represent the critiques, i.e., adopts both unit critique and compound critiques [7] and which critiques to present for users, i.e., only presents available critiques based on the candidate items [60]; (2) the critiques-based user preference modeling [129]; (3) grant users the freedom to customize their critiques for more efficient interaction [80]; (4) how to retrieve the items based on the critiques, i.e., according to both the current constraints and the previous critiques [98] [61]; (5) how to design the user interface to improve the efficiency and effectiveness of the interactions [128]. A critique-based system could also have a sequence of interactive graphical user interfaces, for example, [56] models multi-step webpage interactions with the reinforcement learning method. At each step, their model decides whether to collect more information from the user or to show the recommendation page to users.

Comparing to dialogue systems, critique-based systems have their limitations. First, there is overwhelming information in the user interface of the critique based system, and a user has to spend time to get familiar with the system before using it. However, interacting with dialogue systems is a more succinct and easy-to-use procedure. Second, with critique based systems, a user can only manipulate the critiques, which are usually predefined and limited. However, dialogue systems can support much more flexible interactions, for example, free-form natural language utterances.

Recommendation in Conversational Dialogue

Research in this direction is at its early stage. Yan et al. [124] proposed a conversational dialogue recommender framework based on a mobile online shopping application. They formulate the framework into different components, and within each component, they build a separate model to perform the task. By passing and

processing the data among different components, their system functions as a whole for conversational online shopping. Christakopoulou et al. [13] built a system to ask a user to provide opinions on if he likes an item or prefers one item to another repeatedly, based on which their system improves the recommendation results. One can feel bored in such a process if he is asked to compare items constantly. A more natural process is to discuss the attributes of items in natural languages to express opinions. Dodge et al. [19] built an end-to-end dialogue agent that can recommend movies based on the context information using the memory network. Zhang et al. [133] built a conversational search and recommender system using a similar model. By utilizing the review information in amazon dataset, their framework can select facets to ask the user and recommend a list of items accordingly. However, these systems have relatively fixed dialogue discourses and do not have recommendation related decision makings, for example, when to collect more information and when to make a recommendation.

Chapter 3

Conversational Recommendation Dialogue Data Collection

3.1 Overview

A conversational recommender system should act like a real human sales agent to help users find their preferred items. In this procedure, a dialogue agent should guide the user through the process, by strategically asking users appropriate questions to elicit their preferences, and making recommendations accordingly. The agent is the expert and should know which questions to ask. Also, it should know when to stop asking questions because enough information has been collected for making high-quality recommendations. In order to build such a system, we need dialogue datasets that have utterances of exchanging the facet values between users and the dialogue agent. The dialogue should also contain user-item information in order to make personalized recommendations to users. To the best of our knowledge, there are no available public datasets that can be directly used to support the research problem we want to explore. To facilitate the research

in this direction, we build the conversational recommendation dialogue datasets based on popular recommender datasets, using the technique of crowdsourcing [20]. The dialogues contain the utterances between a user and a dialogue agent discussing the facet values of a target item.

3.2 Related Work

According to how the utterances of the dialogue are generated, existing dialogue datasets could be generally categorized into three types [94]:

Machine-Machine Dialogue Dataset: These type of dialogue data are generated via templates or patterns. For instance, Borders et al. [5] used templates and a knowledge base to generate simulated dialogues to build a task-oriented dialogue system. Pure template-based dialogues are usually not natural enough. An important improvement is proposed by [95] and [105]. In these works, a schema is first simulated. Then crowdsourcing workers are requested to write natural language paraphrases based on schemas. Our approach is slightly different from [95] in that our dialogues are collected through an interactive dialogue user interface, rather than through filling web forms, so our method is more consistent with the actual online chatting scenario.

Human-Machine Dialogue Dataset: Dialogues of this type have utterances between real users and an existing dialogue system. An early dataset is the Let’s Go Public corpus [78], which has the conversations of real users interacting with a bus schedule informational dialogue agent. Examples of the more recent datasets are the popular Dialog State Tracking Challenges (DSTC 1-3) datasets [29], which are in the domains of bus scheduling and restaurant booking. This type of dataset usually requires a working dialogue system to let users interact with. The dialogue utterances are more natural than machine-machine dialogues.

However, the quality of the dialogues is constrained by the quality of the existing system.

Human-Human Dialogue Dataset: Dialogues of this type are collected via the human-human conversation. The more recent DSTC 4-7 datasets are collected in this way. There is another type of dataset collected via the Wizard-of-OZ paradigm, in which the subjects are told to interact with a machine, but are in fact interacting with real human beings. One example among the early dataset is the ATIS corpus [28]. More recent ones include [119] [6]. Although this type of dataset has the best naturalness among the three types, it is usually not cost-efficient to collect dataset in this way, especially for our case where there are millions of rating pairs of a typical recommendation dataset. A comprehensive review of the dialogue dataset can be found in [94].

The dialogue datasets can also be categorized based on the type of dialogue system they are used to build. The first type is the general-purpose dialogue corpus, which can be collected from the conversation related scenarios, and they do not have to be in the strict dialogue context. For example, the Ubuntu Dialogue Corpus [54] is a dataset from the online chat room which has two or more participants discussing in real-time about the issues related to the Ubuntu system. [88] collected 1.3 million conversations from the Twitter Microblog. The second type is the information-seeking dialogue dataset, for example, the single round question answering dataset [77], multi-round reading comprehension dataset [79], and the factoid question answering dataset [46]. The third type is the task-orientated dialogue dataset. There are a lot of corpus in this category, spanning various applications and domains, such as movies [19], restaurants [29] and tourism [35]. [19] created a conversational recommendation dataset using the MovieLens Dataset. However, their dataset does not contain the utterances discussing facet values of

the target item.

Another related work is the delexicalization technique [119] [117]. It is a method to abstract natural language utterances into templates by substituting a subfragment with the corresponding placeholders. This approach enables the creation of templates based on the natural language sentences and allows knowledge transferring across different domains.

3.3 Faceted based Recommendation Dataset Processing

We build the conversational dialogue dataset based on two popular recommendation datasets: the Yelp Challenge dataset¹ and the MovieLens1M dataset [24]. These two datasets have abundant user-item explicit rating feedback. Movie and restaurant are two of the popular domains in both the recommendation and the dialogue research communities [18] [119] [29]. We make an assumption [105] that users have selected their preferred items after chatting with a dialogue agent. Though this might be a strong assumption, it is a reasonable starting point for conducting research in this direction. To preprocess the recommendation datasets, we keep users and items that have at least 5 ratings. Also, the Yelp Challenge dataset contains items of various business types, for example, “shopping”, “health”, and “automotive”. We filter these types of items out to keep only items with “restaurant”, “food”, “bars” and “nightlife” business types. Statistics of the dataset after preprocessing are shown in Table 3.1. The two datasets have different levels of sparsity, and the MovieLens1M dataset is much denser one than the Yelp dataset, which has a density of 0.0661%.

¹<https://www.yelp.com/dataset/challenge>

Table 3.1: Basic statistics of the pre-processed MovieLens1M and Yelp recommendation datasets.

	# of User	# of Item	# of Rating	Density
MovieLens1M	6,040	3,412	999,458	4.84%
Yelp	62,047	21,350	875,721	0.0661%

In the next step, we collect the facet values for each item in both datasets, because our conversational recommendation agent will discuss attributes of a target item with users. The Yelp Challenge dataset has already provided rich facet values information. For the MovieLens1M dataset, we use the OMDb Api² to collect movie features from the online movie database IMDb³.

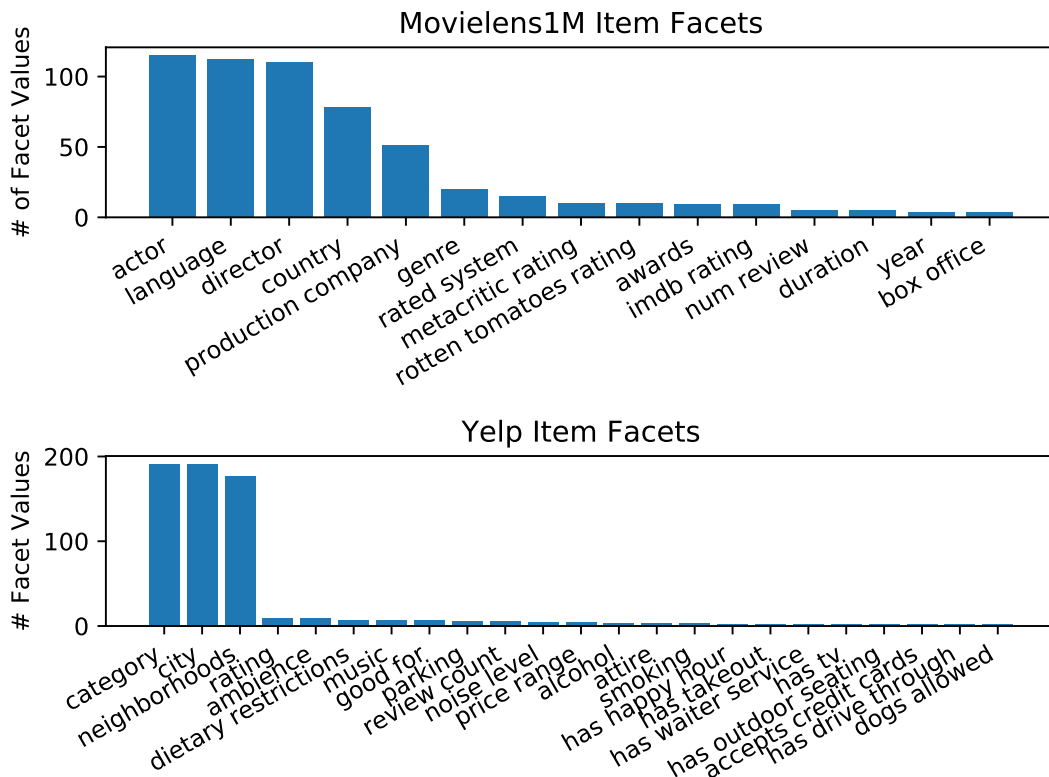


Figure 3.1: The histogram diagram of the number of facet values of items on the MovieLens1M (upper) and Yelp Challenge dataset (lower).

²<http://www.omdbapi.com/>

³<https://www.imdb.com/>

After collecting facet values, we continue to preprocess the values of each facet. For numerical facets of both datasets, we bin their values into nominal values by range. For example, we bin the “review count” facet in the Yelp dataset into ranges of “ ≤ 20 ”, “21-50”, “51-100”, “101-500”, “ ≥ 500 ”. We also filter the nominal facets by frequency. For example, in the MovieLens1M dataset, we filter out the actor and director values, whose frequency is less than 5. Note that the Yelp dataset has several “true/false” facets, for example, the “has tv” and “has taken out”. After processing, there are 15 facets in the MovieLens1M dataset and 23 facets in the Yelp dataset. The numbers of facet values in both datasets are shown in Figure 3.1. To this point, each item in both datasets is augmented with the processed facets values.

3.4 Crowd Sourcing based Conversational Dialogue Dataset Collection

As discussed earlier, there is no publicly available dialogue dataset that can directly support making a personalized recommendation for users based on the discussions of item facet values. We build two datasets following the procedures in [95] and [105], with the help of crowdsourcing techniques.

The ideal dialogues would be ones having a real user first chat with a dialogue agent to inform the facet values of a target item and then select this item from a list of items recommended by the dialogue agent. However, it is not possible for us to collect such dialogues from real users who generate the ratings in the original recommendation dataset. Besides, collecting dialogues with human-human or human-machine conversations is less cost-effective, especially for recommendation datasets, which have millions of rating information. Moreover, dialogues between

Table 3.2: An example of a simulated dialogue schema for a user item pair in the MovieLens1M dataset

The target movie has the following facet values: {genre: Drama, year range: before 80s, language: French, rating range: 8.0-9.0}
<u>User</u> : inform(rating="8.0-9.0")
<u>Agent</u> : request(genre)
<u>User</u> : inform(genre="Drama")
<u>Agent</u> : request(director)
<u>User</u> : dontknow()
<u>Agent</u> : request(year_range)
<u>User</u> : inform(year_range="before 80s")
<u>Agent</u> : request(language)
<u>User</u> : inform(language="French")
<u>Agent</u> : recommend()
<u>User</u> : thanks()

two real users tend to have the “reasonable” discourses [95]. In other words, the discourse can only cover a limited percentage of “dialogue paths” [32]. However, in order to train a robust dialogue agent, we need to show as many dialogue paths as possible to the dialogue agent, even ones that are logically incorrect [95]. Based on these considerations, we resort to the simulation-based approach to generate dialogues.

We mainly collect the user utterances in the dialogue. For machine utterances, we sample them from a prescribed candidate set. First, we random simulate the dialogue schemas [95] [105], which are template like sentences. A schema contains an intention, facets and their values (if any) [120] [106]. For example, a schema “inform(genre=“Drama”)” denotes “to inform the genre of the movie is Drama”. A schema “request(rating)” denotes “to request the rating of the movie”. We used four types of schemas: “request(facet)”, “inform(facet=value)”, “dontknow(facet)” and “thanks()”. At each turn, the dialogue agent requests one or more facets from a user and the user either inform the values within a response

if she knows it, or tells the agent she does not know it. The dialogue ends with the agent making a recommendation to the user. Table 3.2 shows an example of a simulated dialogue schema.

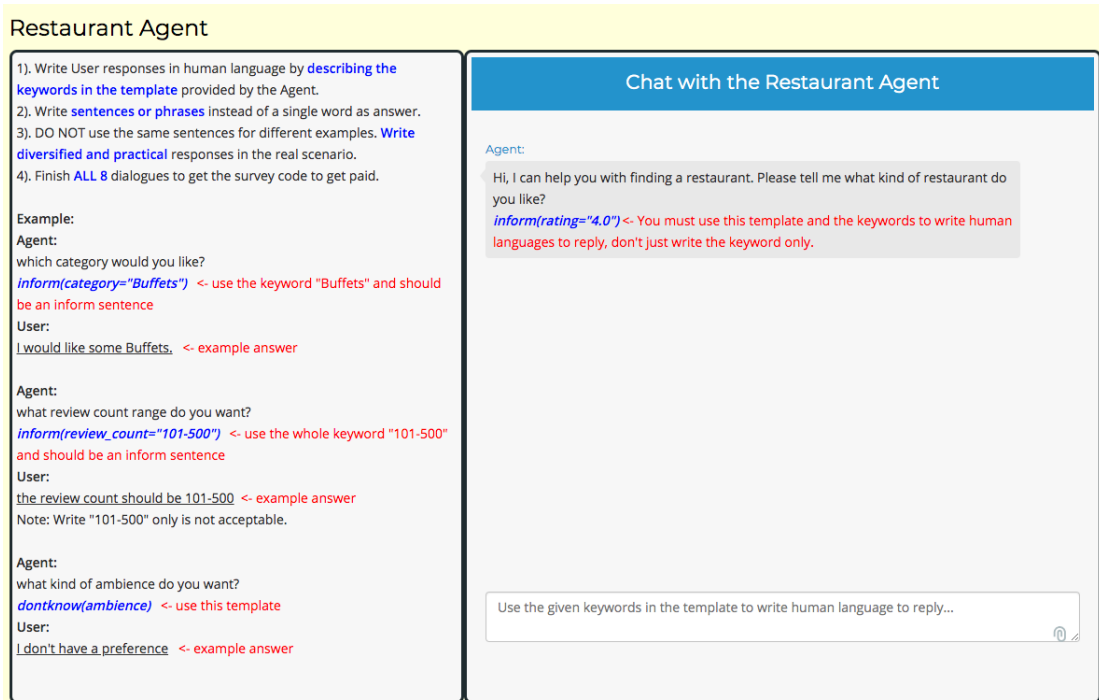


Figure 3.2: The dialogue collection graphical user interface. On the left panel shows instructions and several examples. On the right panel is the dialogue window, where an MTurk is requested to paraphrase the schema into natural language user utterance.

We use the crowdsourcing service provided by the Amazon Mechanical Turk (AMT) website⁴. Only qualified workers are hired to do our tasks. In the HIT (the task), MTurks (workers in the AMT) are first given an instructional overview. Then they are presented with the graphical user interface, as shown in Figure 3.2 to do the task. On the left part, they are given several examples for different types of schemas and the corresponding example utterances. The right part is a dialogue chatting window. At each turn, an MTurk sees a dialogue agent utterance,

⁴<https://www.mturk.com/>

Table 3.3: The crowdsourced dialogue scripts using the schema in Table 3.2

The target movie has the following facets: {genre: Drama, year range: before 80s, language: French, rating range: 8.0-9.0}
<u>User:</u> the rating should be 8.0-9.0
<u>Agent:</u> which genre would you like?
<u>User:</u> i would like Drama movies
<u>Agent:</u> who are you favourite director?
<u>User:</u> Don't care.
<u>Agent:</u> which year range would you like to watch
<u>User:</u> i like before 80s year movies
<u>Agent:</u> which language would you like?
<u>User:</u> French
<u>Agent:</u> recommend you with the following items
<u>User:</u> Thanks

accompanied by a schema of the user response. The Mturk needs to paraphrase this schema into a natural language utterance. One must address all the facet and values (if given) in the schema. We require Mturks to write responses in natural human language and encourage them to write diverse sentences. An example of a written dialogue is shown in Table 3.3, which is a scripted dialogue by an Mturk using the schema in Table 3.2.

After collecting the user utterances, we perform an additional round of MTurk experiments to filter out low-quality sentences. Specifically, for each written utterance and its paired schema, we request qualified MTurks to label whether or not the utterance correctly paraphrases the schema with natural language and if it has addressed all the facet value in the schema. We only keep the ones that are given positive labels by two or more MTurks. After this round of experiments, we collect a set of high-quality sentence level user utterances.

Following the approach in [119], we convert these utterances into templates. Specifically, we use string matching to locate a subfragment in the MTurk written

utterances and replace them with placeholders. This is a process of delexicalization defined by [119]. Following this process, we get a set of sentence-level templates. For example, an utterance “I want to watch a Comic Movie” is transformed into the template “I want to watch a *<GENRE>* movie”, in which *<GENRE>* is the placeholder. Then we use these natural language templates to simulate the conversational recommendation dialogues randomly. Each dialogue has a target user-item pair, a conversation discussing the facet values of the target item, and facet value labels for each user utterance. These dialogues are used to train natural language understanding and recommender modules. Note that we do not train the dialogue manager with the policy information contained in the collected dialogues. Therefore, the random simulation of the dialogue utterances is acceptable. We train the dialogue manager with a reinforcement learning approach, which is to be introduced in the following chapters.

We generate dialogues for each user-item pair in the preprocessed recommendation dataset. Statistics of the collected dialogue dataset is shown in Table 3.4. We collected 331 user utterance template for the MovieLens1M dataset and 613 templates for the Yelp dataset. The total number of utterances and tokens are large due to the big number of user-item pairs from the recommendation dataset. However, the unique tokens and unique bigram are comparable to the DSTC2 dataset [95]. Two examples of the collected dialogues are shown in Table 3.5. The user utterances are simulated based on the simulated facet value pairs and the delexicalized templates. The machine utterances are sampled from a set of pre-scripted templates. We can observe that the facet values are automatically labeled, which are shown in the parentheses following each user utterance. We make the dataset public available⁵.

⁵<https://github.com/ysun30/ConvRec>

Table 3.4: Statistics of the two conversational recommendation datasets, comparing to the DSTC2 dataset.

	DSTC2	MovieLens1M	Yelp
Dialogues	1611	999,458	875,721
Templates	-	331	613
Users	-	6,040	62,047
Items	-	3,412	21,350
Total utterances	11,670	15,272,360	12,333,121
Total tokens	199,295	101,002,615	75,713,435
Avg. utterances per dialogue	14.49	15.2802	14.0820
Avg. tokens per utterance	8.54	6.6151	6.1386
Unique tokens	977	1,715	2,027
Unique bigram	3,528	8,440	13,461

Table 3.5: Example dialogues simulated using the method discussed in this chapter. The left dialogue is a MovieLens1M example, the right dialogue is a Yelp example.

MovieLens1M	Yelp
<p><u>User:</u> a movie directed by Clint Eastwood and in 90s <i>(director="Clint Eastwood", year="in 90s")</i></p> <p><u>Agent:</u> which production company of the movie do you like?</p> <p><u>User:</u> movies produced by warner brothers <i>(production="warner brothers")</i></p> <p><u>Agent:</u> which genre would you like?</p> <p><u>User:</u> i would like the genre to be Mystery <i>(genre="Mystery")</i></p> <p><u>Agent:</u> which language would you like?</p> <p><u>User:</u> i want a movie in English <i>(language="English")</i></p> <p><u>Agent:</u> do you want movies with any awards?</p> <p><u>User:</u> has some awards or nominations <i>(award="other win or nomination")</i></p> <p><u>Agent:</u> which rating range would you like?</p> <p><u>User:</u> i like movies with a rating between 6.0 to 7.0 <i>(rating="6.0-7.0")</i></p> <p><u>Agent:</u> which actor do you like?</p> <p><u>User:</u> I would like to watch a movie with Clint Eastwood <i>(actor="Clint Eastwood")</i></p>	<p><u>User:</u> i need a rating of 3.5 restaurant <i>(rating="3.5")</i></p> <p><u>Agent:</u> Which city are you in?</p> <p><u>User:</u> i'm in Tempe <i>(city="Tempe")</i></p> <p><u>Agent:</u> any preference for the parking site?</p> <p><u>User:</u> i would like lot parking. <i>(parking="lot")</i></p> <p><u>Agent:</u> which review count range do you want?</p> <p><u>User:</u> the review count should be between 101 and 500 <i>(review_count="101-500")</i></p> <p><u>Agent:</u> do you want one that accept credit card?</p> <p><u>User:</u> sure <i>(accept_card="True")</i></p> <p><u>Agent:</u> which category do you want?</p> <p><u>User:</u> i want a place that has Cupcakes <i>(category="Cupcakes")</i></p> <p><u>Agent:</u> which price range do you want?</p> <p><u>User:</u> a medium price range. <i>(price_range="medium")</i></p> <p><u>Agent:</u> recommend you with the following items</p> <p><u>User:</u> Thanks a lot!</p>

3.5 Summary

In this chapter, we introduced the conversational recommendation dataset collected via the crowdsourcing technique. We argue that in order to make good recommendations to users, the dialogue agent should chat with a user to understand the information need towards target items and make recommendations accordingly. Existing datasets do not meet our requirements as the existing recommendation datasets do not have the associated dialogues scripts, and the existing dialogue datasets do not contain the personalized user-item information needed for making personalized recommendations, or the utterances discussing facet values of target items.

We build the dataset based on two recommendation datasets. We collected facet values of the target items and simulated dialogue schemas for each user-item pair in the dataset. Then we request the crowdsourcing workers to paraphrase schemas into natural language utterances addressing all the facet values in a schema. We further delexicalize the collected user utterances into sentence-level templates. Finally, we simulate the conversational recommendation dialogues using these templates. In the next chapters, we will present how the collected datasets are used to train different modules in our proposed conversational recommendation framework. We hope these datasets can facilitate researchers who are also interested in this field.

Chapter 4

The Framework of Conversational Recommender System

4.1 Overview

In this chapter, we discuss how to build a conversational recommender system. The goal of the system is to recommend personalized item(s) to users successfully, and it achieves this goal by analyzing what the user has said in the current session, interactively asking user clarification questions, and making personalized recommendations when appropriate, based on the current session and what the user has consumed or rated before. Our framework has three components: a belief tracker, a recommender system, and a dialogue manager.

Several aspects are important in the process. First, the dialogue agent needs to understand the user's intention correctly. Second, the agent should make sequential decisions and take appropriate actions in each turn. Third, the system needs to make personalized recommendations in order to maximize user satisfaction. Figure 4.1 illustrates an overview of our proposed framework. At a time

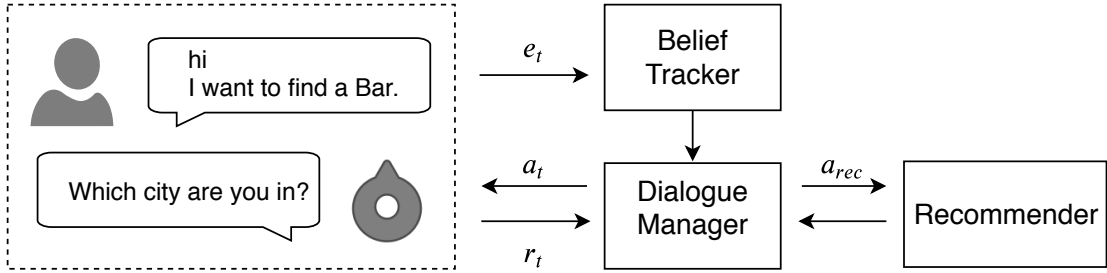


Figure 4.1: The conversational recommender system overview

step in the dialogue, the user utters “I want to find a Bar”. The framework calls the belief tracker to convert the utterance into a vector representation or “belief”; then the belief vector is sent to the dialogue manager for the dialogue agent to make a decision. For example, the agent may decide to request the city information next. Then the agent responds with “Which city are you in?”, and gets a reward, which is used to train the dialogue policy. A different decision is to make a recommendation. Then the agent calls the recommender module to get a list of items personalized for the user. We introduce each component and the relationships among them in more details in the following sections.

4.2 The Conversational Recommender System Framework

4.2.1 Belief Tracker

When try to buy products on an e-commerce website, users often navigate the product space through faceted search [130][36][113]. Motivated by this and in order to assist users to find the item they want in conversation, it is crucial that the system understands which values the user has provided for product facets, and represents the user utterances with a semi-structured query. We introduce a Belief

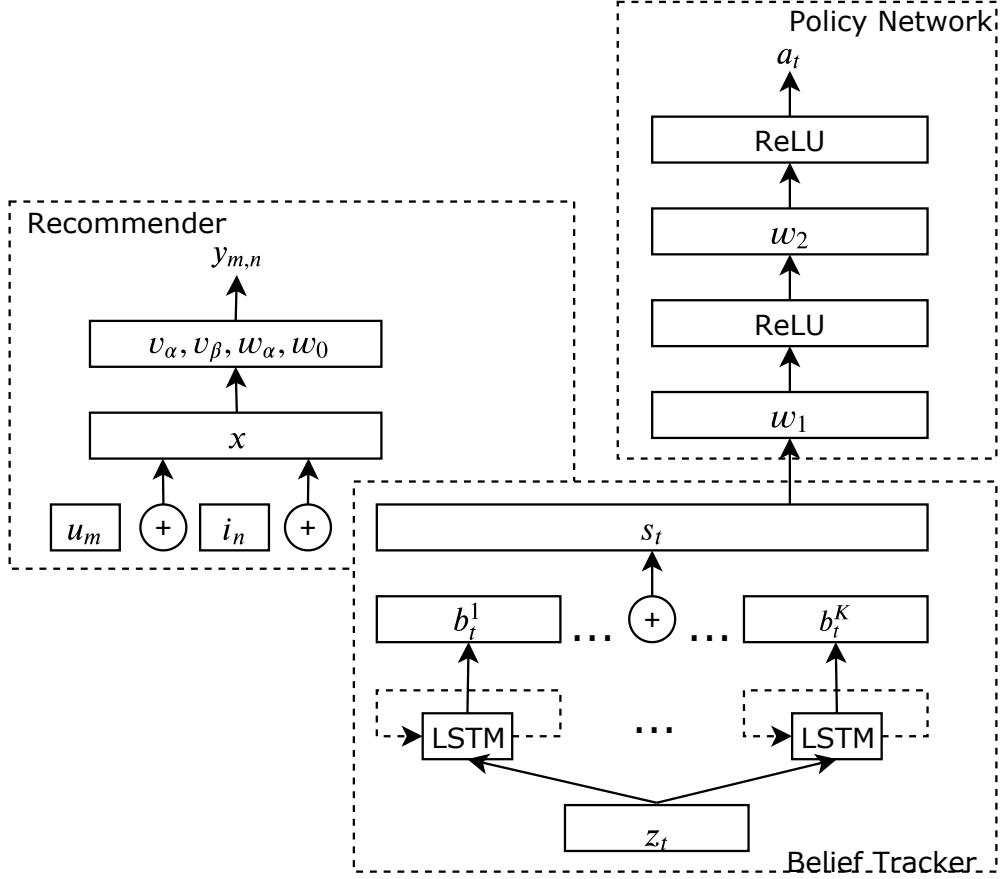


Figure 4.2: The structure of the proposed conversational recommender framework. The bottom part is the belief tracker, the top left part is the recommendation model, and the top right part is the deep policy network.

Tracker module similar to [18] to extract facet-value pairs from user utterances during the conversation, and maintain the facet-value pairs as the memory state (i.e., user query) of the agent. In this dissertation, we view the product facet (or attribute, metadata) f along with its specific value v as a facet-value pair (f, v) . Each facet-value pair represents a constraint on the items. For example, $(color, red)$ is a facet-value pair which constrains that the color of the items need to be red.

The network structure of belief tracker is shown in the lower part in Figure 4.2. We train a belief tracker for each facet of the items. The belief tracker takes the

current and the past user utterances as the input, and outputs a probability distribution on all the possible values of a facet at a time step. The dialogue system’s belief of the session is constituted by the predicted values of all the facets. Specifically, given a user utterance at time step t , e_t , the input to the belief tracker is the n-gram vector z_t . Let V denotes the n-gram vocabulary, thus $z_t \in \mathbb{R}^{|V| \times 1}$.

$$z_t = \text{ngram}(e_t) \quad (4.1)$$

Suppose there are K types of facets, we denote a facet f_m , with $(1 \leq m \leq K)$. At step t , the sequence of n-grams up to the current time z_1, z_2, \dots, z_t is encoded by a Long Short Term Memory (LSTM) [33] into a vector $h_t \in \mathbb{R}^{|f_m| \times 1}$, where $|f_m|$ is the number of values of f_m . The hidden vector h_t is then fed to a softmax activation layer to be transformed to the probability distribution $b_t^m \in \mathbb{R}^{|f_m| \times 1}$. For each facet, we train an LSTM separately.

$$\begin{aligned} f_t &= \sigma_f(W_f z_t + g_f) \\ o_t &= \sigma_o(W_o z_t + g_o) \\ i_t &= \sigma_i(W_i z_t + g_i) \\ c_t &= f_t \odot c_{t-1} + i_t \odot \tanh(W_c z_t + g_c) \\ h_t &= o_t \odot \tanh(c_t) \end{aligned} \quad (4.2)$$

where the parameters W_f, W_o, W_i, W_c belong to $\mathbb{R}^{|f_m| \times |V|}$, and the bias g_f, g_o, g_i and g_c belong to $\mathbb{R}^{|f_m| \times 1}$. At round t , all the b_t^m are concatenated together to form the agent’s current belief of the dialogue state in the current session.

$$b_t = b_t^1 \oplus b_t^2 \dots \oplus b_t^K \quad (4.3)$$

where b_t has the dimension of $\sum_{m=1}^K |f_m|$. By using the learned output of the

LSTM network directly, we keep the uncertainty from the belief tracker for the following modules. This vector has the information that which values of which facets have been addressed in the dialogue so far.

4.2.2 Recommender Module

As the conversational system interacting with the users, at certain round, the dialogue agent can decide to make a recommendation based on its current belief of the user’s information need, which is represented by the belief vector. We train a recommender using the belief vector, the user information, and the item information. Specifically, we use the Factorization Machine (FM) [81], for the reason that FM can easily incorporate additional features, e.g., s_t , for the model training.

The structure of the recommendation model is shown in the upper left part of Figure 4.2. Let u denote the users and i the items. For M users and N items in the dataset, the users and items are represented as the sets: $\{u_1, u_2, \dots, u_M\}$ and $\{i_1, i_2, \dots, i_N\}$. The input x to the recommender is the concatenation of the 1-hot encoded user vector and item vector, where the only element that is not zero in the vector corresponds to the index of a user or an item, and the dialogue belief vector b_t :

$$x = u_m \oplus i_n \oplus b_t \tag{4.4}$$

where m and n are the indexes of a user and an item, and

$$u_m = \{0, 0, \dots, 1, \dots, 0\}, \text{ with } 1 \text{ at the } m_{th} \text{ element.} \tag{4.5}$$

$$i_n = \{0, 0, \dots, 1, \dots, 0\}, \text{ with } 1 \text{ at the } n_{th} \text{ element.} \tag{4.6}$$

The output $y_{m,n}$ can be either explicit feedback, e.g., a rating score, or implicit

feedback, e.g., a 0-1 scalar. We use a 2-way ($K = 2$) FM [81]:

$$y_{m,n} = \mathbf{w}_0 + \sum_{\alpha=1}^N \mathbf{w}_\alpha x_\alpha + \sum_{\alpha=1}^N \sum_{\beta=\alpha+1}^N \langle \mathbf{v}_\alpha, \mathbf{v}_\beta \rangle x_\alpha x_\beta \quad (4.7)$$

$$\langle \mathbf{v}_\alpha, \mathbf{v}_\beta \rangle = \sum_{\kappa=1}^K \mathbf{v}_{\alpha,\kappa} \mathbf{v}_{\kappa,\beta} \quad (4.8)$$

where \mathbf{w}_0 , \mathbf{w}_α , \mathbf{v}_α and \mathbf{v}_β are learnable parameters. α and β denote the index of elements in the input vector x , and $y_{m,n}$ is the rating score given to i_n by u_m . For rating prediction, Stochastic Gradient Descent (SGD) is used to minimize the square loss between the predicted rating score and the real rating score.

$$Loss(\Theta) = \sum_{m=1}^M \sum_{n=1}^N (y_{m,n} - y)^2 \quad (4.9)$$

where Θ denotes all the parameters, and y is the real rating. Note that, according to [81], the objective function scales linearly with the size of the data.

Without loss of the generality, when the dialogue agent needs to make recommendations using the trained recommender, it first takes the *argmax* of each facet’s belief vector, to get K categorical distributions over the values, one for each facet. The combinations of the facet values form a new distribution, with the probability the product of K value’s probabilities. We keep the μ most probable combinations and use their facet value combinations to retrieve items from the entire item set. The retrieved items form a candidate set. Then we use the trained model to re-rank the candidates based on their rating scores. Note that due to the belief tracking errors, e.g., misunderstanding the user’s intention, the target item may not exist in the candidate set.

4.2.3 Deep Policy Network based Dialogue Manager

The dialogue manager is a crucial component of the dialogue system. It decides which actions to take at each dialogue turn. Reinforcement learning based agent aligns well with a dialogue agent because reinforcement learning is suitable for problems involving sequential decision-making [109]. We train a deep policy network to act as the dialogue manager, i.e., at each dialogue turn, it decides which action to take, in order to maximize the long-term expected return. It looks at not only the short term reward, or immediate reward, but also the long term accumulative rewards.

The policy gradient method can learn a policy directly, without consulting the value functions [109]. One of the advantages of the policy gradient methods over the action value methods is that it has stronger convergence guarantees. There are pros and cons of the value-based methods and the policy network methods. Readers may refer [109] for more details.

The structure of the policy network is shown in the upper right part in Figure 4.2. Following the Markov Decision Processes (MDPs), the policy gradient model has the five tuples $\{s_t, a_t, r_t, \pi(a_t|s_t), \gamma\}$.

State: The state s_t is a description of the current environment that the agent is interacting with. In our case, we define it as the conversation context, which is the belief vector, $s_t = b_t$.

Action: At each time step t , the agent needs to select an action a_t based on the information of the state s_t . In our setting, we have mainly two types of actions. One type is to request the value of a facet, which is further divided into K actions $\{a_1, a_2, \dots, a_K\}$, one action per facet. The other is to make a personalized recommendation a_{rec} , in which case the recommendation module described above would be called. Note that a_{rec} may in fact occur more than once in a single

conversation session. We leave the modeling of multiple recommendations in one dialogue session for future work.

Reward: The reward is the benefit or penalty the agent gets from interacting with an environment. At each turn, according to the current state s_t , the agent selects an action a_t following the policy, and gets an immediate reward r_t , denoting how good the current decision is. Then the state s_t transits to a new state s_{t+1} . In our framework, the dialogue manager gets a reward when it requests a facet value, or makes a recommendation. Note that the reward is the feedback from the environment. The recommender module serves as part of the environment. The agent only gets rewards from the environment but can not change it [109]. We model the recommendation reward in different ways, which will be introduced in section 4.3.1.

Policy: The conversational recommendation scenario is an episodic process where the whole dialogue is an episode. The policy is denoted as $\pi(a_t|s_t, \theta)$, which is the probability of taking action a_t when the agent is in state s_t . We use a deep neural network as the function approximator, because the states are in continuous space, and it is not possible to enumerate all of them. For simplicity and without loss of generality, our policy network contains two fully connected layers, and each has an *ReLU* activation function. Other neural network structures may also work. The output of the network is further sent to a softmax layer to be transferred to the probability of the actions. Based on the policy gradient theorem, the REINFORCE algorithm [122] is derived as

$$\theta_{t+1} = \theta_t + \alpha G_t \frac{\nabla \pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta)} \quad (4.10)$$

This is the target the model tries to learn. Note that the quantity $G_t \frac{\nabla \pi(a_t|s_t, \theta)}{\pi(a_t|s_t, \theta)}$ is a sampled quantity at each timestep whose expectation equals to the gradient

of the performance measure $J(\theta)$ that the method tries to maximize:

$$\nabla J(\theta) = E_{\pi} \left(G_t \frac{\nabla \pi(a_t | s_t, \theta)}{\pi(a_t | s_t, \theta)} \right) \quad (4.11)$$

where θ is the policy parameter to be learned. Here G_t is the discounted sum of the total episodic rewards, or return, starting from time step t to the final time step T :

$$G_t = \sum_{t=0}^T \gamma^t r_t \quad (4.12)$$

with γ the discount parameter that controls the weight of the immediate reward in return.

In our case, we always have a terminating state of the conversation, e.g., the user leaves the chat, or the user is successfully recommended with a target. This enables us to use gradient ascent methods to optimize the parameter θ in the policy network directly. Note that if θ is initialized randomly, the learning can fail completely. To address this issue, we use a rule-based policy, which is introduced in the next section, to initialize the parameters. The policy gradient algorithm is presented in Algorithm 1.

4.3 Experimental Setup

4.3.1 User Simulation

It is common practice in the reinforcement learning field to first simulate an environment, in our case is a simulated user, to interact with the agent. User simulation has several advantages [48]: (1) allows the reinforcement learning agent to explore different trajectories that may not be covered in the real user interaction data. (2) it is a more cost-effective approach than running the reinforcement

Algorithm 1 The learning process of the policy gradient agent.

```
1: for each data in training batch do
2:    $t = 0$ , set episode set  $\mathcal{E}=\{\}$ 
3:   initiate a dialogue with starting utterance  $e_t$ 
4:   while not dialogue ends do
5:     Apply the belief tracker to get the belief vector  $b_t$ 
6:      $s_t = b_t$ 
7:     Sample  $a_t$  from the policy network  $\pi(a_t|s_t, \theta)$ 
8:     Execute  $a_t$  to get  $r_t$  and user response from the user simulator
9:     Add  $\langle s_t, a_t, r_t \rangle$  to  $\mathcal{E}$ 
10:    If  $a_t$  is recommend, then dialogue ends
11:     $t = t + 1$ 
12:   $G_t = 0$ 
13:  for step  $t$  in episode do
14:     $G_t = \sum_{k=t}^T \gamma^{k-t-1} r_k$ 
15:    update  $\theta$ :  $\theta = \theta + G_t \nabla \ln \pi(a_t|s_t, \theta)$ 
```

learning model online to learn from interacting with real users to learn the policy. (3) learning from the simulated agent instead of learning from scratch with the real users prevents the system from hurting user satisfaction, which is caused by the incompetence of the system at its early stage.

Simulation of the user dialogue agent interaction

We assume that the user is cooperative while interacting with the dialogue agent. The user has three behaviors: (1) if the system requests the facet values, the user will respond with a natural language utterance containing the facet values; (2) if the system recommends a list of items, the user will examine the list to find the target item; and (3) the user will leave the dialogue if one of the following cases happens: i) the dialogue is too long; ii) the target item is not in the recommendation list; iii) the target item is in the list but is ranked too low. In the simulation, we assume that the user always knows which item she wants. Note that this might not be true in the real case. As the user may leave at any

time, or may not be cooperative.

In each of the above situation, the user gives a numerical reward to the dialogue agent. A small negative reward r_c is given to the agent if the user answers the agent’s question, and the dialogue continues. The r_c is to prevent the dialogue from getting too long. The agent receives a positive reward r_p if the target item is in the recommendation list and ranked high, e.g., above a threshold. A negative reward r_q is given if the user quits the conversation. We summarize the user behaviors in Algorithm 2.

Algorithm 2 The interaction between the agent and the simulated user

```

1: Start with M epochs, N training data
2: for epoch = 1, M do
3:   for i = 1, N do
4:      $t = 0$ 
5:     Sample a  $(u, i)$  pair from the training set
6:     The user  $u$  starts the conversation, and conveys a random facet value
       with utterance  $e_t$ 
7:     while True do
8:       Apply belief tracker to  $e_t$  to get  $s_t$ 
9:       Send  $s_t$  to the policy gradient agent
10:      Get action  $a_t$  from the agent
11:      if  $a_t$  is  $a_{rec}$  then
12:        Call the recommender to get an item ranking list
13:        if the target item is in the top K then
14:           $r_t = r_p$ , the dialogue succeeds and break
15:        else
16:           $r_t = r_q$ , the dialogue fails and break
17:      else
18:        The system generates a response  $e_m$  to the user
19:        if the user quits then
20:           $r_t = r_q$ , the dialogue fails and break
21:        else
22:           $r_t = r_c$ ,  $t = t + 1$ 
23:          The user responds with a new utterance  $e_t$ 

```

Simulation of user behavior of checking the recommendation results

When recommended with a list of items, the user has different ways of examining the results. We set a stopping threshold K for the recommendation list, beyond which the user stops checking. We also denote C as the maximum positive reward a user can get, e.g., $r_p \leq C$. The three user behaviors are:

Linear r_p : the user always checks the next item in the list with the probability of 1, until the target item is found, or the end of the list is reached or the stopping threshold K is reached. In this case, the reward is defined as: $r_p = \frac{C(K-\tau+1)}{K}$, where τ is the ranking of the target item.

NDCG r_p : following the assumption of NDCG [58], this behavior assumes that users pay more attention to the top positions of the list than the lower positions. Here we use the binary score for the relevant feedback. Thus, $r_p = C * NDCG@K$.

Cascade r_p : following the assumption in [15], the user would examine the list page by page. This is for cases where the dialogue system has a limited user interface, in which case, only a limited number of items can be presented at a time. And the user can navigate among the pages to view more items. In each page, the user views the items equally. This is like a carousel mode. We denote the number of items on a page by κ . The probability of seeing the next page decreases exponentially, with the probability α_1 , and the success reward also decreases exponentially with factor α_2 , where $0 \leq \alpha_1, \alpha_2 \leq 1$. Thus we have $r_p = C * \alpha_2^\rho$, where ρ is the page number and $\rho \leq \lceil K/\kappa \rceil$.

4.3.2 Baselines and Evaluation Methodology

In this chapter, we use the Yelp Challenge dataset with 5 facets: the category, state, city, price range, and rating. In order to evaluate our proposed framework, we compare the model against a set of rule-based baselines. Specifically, we use the

maximum entropy-based baseline, in which the agent computes the entropy of the unknown facet values and always picks the one with the maximum value entropy to request from the user. It stops asking when either the dialogue has reached the maximum turns or all the facet values are known, in which case it recommends a list of items. Therefore, the rule-based policy is a greedy method. We also test a set of its variants, for example, the agent stops to make a recommendation when there are k facets known, with $1 \leq k < 5$. We name it MaxEnt Full for the one asking all the facets and MaxEnt@K for the ones asking exactly k facets. We also compare to one more baseline, which stops asking facet when there is no facet value uncertainty in the candidate itemsets. We denote this approach MaxEnt.

For the evaluation metrics, we use the widely used ones in the dialogue system research. Which are the average return of the dialogues,

$$R = \overline{\sum_{t=1}^T r_t} \quad (4.13)$$

the average dialogue turn,

$$T = \overline{\text{dialogue turn}} \quad (4.14)$$

and the success rate. Note that in recommender systems, the success rate is viewed as the conversion rate, e.g., the percentage of successful recommendations, which is one of the important measures a business tries to maximize:

$$S = \frac{\# \text{ successful dialogues}}{\# \text{ dialogues}} \cdot 100\% \quad (4.15)$$

4.3.3 Model Training Details

For simplicity and scalability, we use a bag of 2-gram representation for the collected user utterances, and the vocabulary size is 19644. To train the belief tracker, we split the entire dataset to a train, dev, and test set with the 80%, 10% and 10% split. We use gradient descent with a learning rate of 0.001 to train the belief tracker. Then we fix the belief tracker’s parameters to train the FM recommender and the policy network. We use the 2-way FM model, with the Adam optimizer and the learning rate of 0.001. For the policy gradient network, we pre-train it as a classifier, by taking the dialogue state as input and the max entropy method’s actions as the labels. After the classifier’s accuracy is stable, we keep on training the policy network using the REINFORCE algorithm [122]. We randomly sample 35000 training dialogues and 26000 dev and test dialogues from the train, dev and test sets, respectively. We use the RMSProp optimizer with a learning rate of 0.001 and the batch size of 100. The best belief tracker that we use has an accuracy of 85.0%. For the reward signals, unless specifically pointed out, we train the simulated users with Linear r_p . We set $r_c = -1$, $r_q = -10$ and the max reward constant C to 40. The discount rate γ is 0.95. The stop threshold K is 30. The maximum dialogue length limit is set to 7. The impact of C and K are shown below.

4.4 Experiment Results

4.4.1 Offline Experiments

Reinforcement learning vs greedy methods

First, we explore how the reinforcement learning method performs in the sequential decision making for the conversation recommender system. Table 4.1 shows the experiment results of the Conversational Recommender Model (CRM) and the baselines on several key measurements. The result of CRM is generated by first training the RL agent and selecting the best model, then evaluating on the test set. Comparing CRM and the MaxEnt model, we observe that the RL agent finds a better policy. In general, the CRM agent is significantly better than all the baselines, in that it can reach a higher average return in shorter average turns, and with higher success rate.

For the baseline methods, we further run several MaxEnt@K methods with $K < 5$ and find that the reward increases as K increases. When $K = 5$, the baseline is MaxEnt Full, and it outperforms the MaxEnt@K methods. We do not expect it to be generally true, especially if K could be very big, such as 100. We also find that MaxEnt is the most competitive baseline method. Thus we use it as the baseline for the following experiments.

We also evaluated the performance while not using the s_t information, i.e., directly recommend items at the beginning of the dialogue. In this case, the model falls back to the plain FM model and performs extremely poor. We find that s_t contributes to the candidate selection step in the recommendation module. However, it does not seem to boost the FM model.

The last two columns of Table 4.1 are also aligned with our expectation. When $k \geq 4$, CRM has a lower “Wrong Quit Rate” because it interacts with the NLU

Table 4.1: Comparisons of CRM and the baselines on Average Return, Average Turn, Success Rate, Wrong quit rate, and Low rank rate. Results in bold are significantly better than baselines at 0.01 level.

	Avg. Return	Avg. Turn	Success %	Wrong Quit %	Low Rank %
MaxEnt@1	-7.0777	1.0002	0.0879	0.0328	0.8793
MaxEnt@2	-3.2687	1.4344	0.2070	0.0460	0.7469
MaxEnt@3	9.4093	2.2524	0.5449	0.0802	0.3749
MaxEnt@4	18.4624	2.9759	0.7544	0.0957	0.1498
MaxEnt Full	21.4557	4.3957	0.8199	0.1027	0.0767
MaxEnt	22.4986	3.8068	0.8251	0.1022	0.0722
CRM	22.6694	3.7724	0.8337	0.0880	0.0783

module less. When $k \leq 3$, the MaxEnt@K baselines have shorter “Wrong Quit Rate” because they interact with the belief tracker much less and perform poorly. For the “Low Rank Rate”, CRM is slightly higher than the MaxEnt method because it is prone to make a faster recommendation and do not narrow the constraints as deep as the latter. Thus CRM’s candidate list is often longer than the MaxEnt method, which makes it more challenging to rank the target item above the threshold. We have similar observations for other MaxEnt@K methods. We also examine the number of candidates in the last recommendation step for each mode. CRM generally has more candidates than the MaxEnt method because MaxEnt adds more constraints than CRM.

Detailed analysis

The CRM can learn that in certain cases, it is better to recommend right away without exhausting all the facet values. This is crucial as users may get bored if they have to provide all the facets before seeing any recommendation. This contributes to CRM’s shorter average dialogue length. Table 4.2 is an example of this case.

Table 4.2: The dialogue with CRM agent (left) is more aggressive than the one with the MaxEnt agent (right).

CRM	MaxEnt
<u>User:</u> i am in the state of AZ	<u>User:</u> i am in the state of AZ (state="AZ")
<u>Agent:</u> Which rating range do you want?	<u>Agent:</u> Which category would you like?
<u>User:</u> 4.5 or above (rating_range>="4.5")	<u>User:</u> i would like Breakfast & Brunch restaurant (category="Breakfast & Brunch")
<u>Agent:</u> Which category would you like?	<u>Agent:</u> Which city are you in?
<u>User:</u> i would like Breakfast and Brunch restaurant (category="Breakfast & Brunch")	<u>User:</u> i am from Scottsdale (city="Scottsdale")
<u>Agent:</u> Which city are you in?	<u>Agent:</u> Which rating range do you want?
<u>User:</u> i am from Scottsdale (city="Scottsdale")	<u>User:</u> 4.5 or above (rating_range>="4.5")
<u>Agent:</u> <recommend, target ranked 3 of 9>	<u>Agent:</u> Which price range do you want?
	<u>User:</u> in a medium price range (price_range>="medium")
	<u>Agent:</u> <recommend, target ranked 4 of 11>

In some other situations, the dialogue failures are caused by the imperfection of the belief tracker component. There are two cases. One case is when the rule-based method fails early in the dialogue turns. Because the belief tracker misunderstand the user’s need and there are no candidates fulfill the constraints. However, the CRM agent continues to ask more facets, which might “bypass” this misunderstanding status and lead to a successful dialogue. Table 4.3 shows an example. Note that the baseline has a belief tracker error in the second and the third round.

The other case is associated with the long dialogue. Again the rule-based method fails due to belief tracker errors. However, the CRM agent does not always take the facet with the maximum entropy, which does not encounter a belief tracker error and succeeds with a shorter turn. In this case, the RL agent takes a different facet trajectory. This case falls into the category of shorter dialogues, as discussed above. An example is shown in Table 4.4. Note that the rule-based agent has a belief tracker error in the fourth turn.

Table 4.3: The MaxEnt agent (right) fails early due to the belief tracker error, but the CRM agent (left) is able to by pass the failure case and succeed.

CRM	MaxEnt
<u>User:</u> i would like Fast Food restaurant <i>(category="Fast Food")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.0 or above <i>(rating_range ≥ "4.0")</i> <u>Agent:</u> Which city are you in? <u>User:</u> i am from Goodyear <i>(city="Goodyear")</i> <u>Agent:</u> Which price range do you want? <u>User:</u> in a cheap price range <i>(price_range>="cheap")</i> <u>Agent:</u> <recommend, target ranked 1 of 2>	<u>User:</u> i would like Fast Food restaurant <i>(category="Fast Food")</i> <u>Agent:</u> Which city are you in? <u>User:</u> i am from Goodyear <i>(no facet found)</i> <u>Agent:</u> Which city are you in? <u>User:</u> i am from Goodyear <i>(state="NV", city="Goodyear")</i> <u>Agent:</u> <recommend, no candidates found>

Table 4.4: The MaxEnt agent (right) fails due to the belief tracker error, but the CRM agent (left) asks a different facet at the second turn, avoids the belief tracker misunderstanding, and succeed with shorter turns.

CRM	MaxEnt
<u>User:</u> I would like Ice Cream & Frozen Yogurt restaurant <i>(category="Ice Cream & Frozen Yogurt")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.0 or above <i>(rating ≥ "4.0")</i> <u>Agent:</u> Which city are you in? <u>User:</u> i am from Henderson <i>(city="Henderson")</i> <u>Agent:</u> <recommend, target ranked 1 of 9>	<u>User:</u> I would like Ice Cream & Frozen Yogurt restaurant <i>(category="Ice Cream & Frozen Yogurt")</i> <u>Agent:</u> Which city are you in? <u>User:</u> i am from Henderson <i>(city="Henderson")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.0 or above <i>(rating ≥ "4.0")</i> <u>Agent:</u> Which price range do you want? <u>User:</u> in a cheap price range <i>(state="PA", price="cheap")</i> <u>Agent:</u> <recommend, no candidates found>

Table 4.5: Comparisons of CRM and the baseline with different user behaviors of checking the recommendation results, on R (Average Return), T (Average Turn), and S (Success Rate).

Model	NDCG r_p			Cascade r_p		
	R	T	S	R	T	S
MaxEnt	13.2990	3.8068	82.51	25.2893	3.8068	80.18
CRM	13.3865	3.7936	83.30	25.4265	3.7634	80.56

Recommendation rewards

To explore the impact of r_p , we run the experiments with the NDCG r_p and the Cascade r_p as discussed in section 4.3.1. We set $\alpha_1 = \alpha_2 = 0.95$. Results are shown in Table 4.5. Note the Linear r_p results have already been listed in Table 4.1. In all three settings, CRM always finds a better policy, with higher average reward and success rate in shorter turns. Linear r_p and NDCG r_p assume the user checks each item until threshold K , and Cascade r_p assumes that the user may leave at any turn. We also observe that the reward of the Linear r_p is higher than the NDCG r_p . Because NDCG r_p penalizes the rewards non-linearly with the decrease of the ranking. Changing the way of modeling r_p is actually changing the environment for reinforcement learning. The experiment results demonstrate that CRM can consistently outperform the baseline in different settings.

The impact of belief tracker accuracy

In our framework, we train the policy network based on the pretrained belief tracker. To study the impact of belief tracker accuracy, we explore how the two models perform on three major measures while varying the belief tracker accuracy from 50% to 85%.

Figure 4.3 shows the belief tracker’s accuracy has an important impact on the proposed framework. The Average Return and the Success Rate increase as the

belief tracker’s error decreases. The Average Length of MaxEnt decreases a bit as the accuracy grows. For all the cases, CRM is better than the baseline. Especially when the belief tracker’s performance is poor. These comparisons reflect the robustness of the reinforcement learning model.

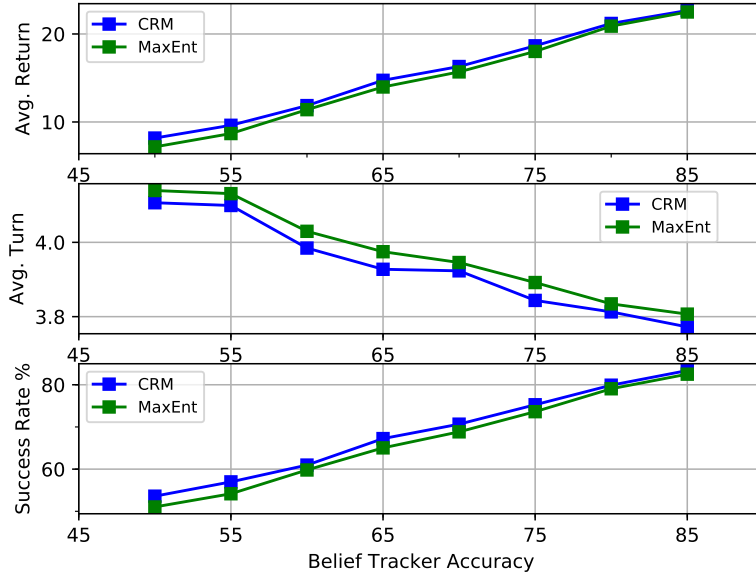


Figure 4.3: Comparisons on the three measures with different belief tracker accuracy.

Different environments

To study the effects of the simulated environments, we vary two major factors, the maximum success reward C , and the recommendation list stop threshold K . Table 4.6 shows how the measures vary with C , and Table 4.7 shows the effects of the threshold K . We observe an increase for the Average Return and the Success Rate as C grows or K increases. In all cases, CRM outperforms the baseline.

With further analysis, we found that the average conversational length of CRM increases with C . This observation is not surprising, because as C increases, it

Table 4.6: Results of R (Average Return), T (Average Turn), and S (Success Rate) with different maximum reward C .

		R	T	S
C=20	MaxEnt	8.9711	3.8068	0.8251
	CRM	9.2028	3.6736	0.8370
C=30	MaxEnt	15.7347	3.8068	0.8251
	CRM	15.9487	3.7674	0.8339
C=40	MaxEnt	22.4986	3.8068	0.8251
	CRM	22.6694	3.7724	0.8337
C=50	MaxEnt	29.2620	3.8068	0.8251
	CRM	29.4842	3.7830	0.8334

Table 4.7: Model performances of three measures with different cutoff threshold K .

		R	T	S
K=20	MaxEnt	19.4010	3.8068	0.7764
	CRM	19.5393	3.7668	0.7827
K=30	MaxEnt	22.4986	3.8068	0.8251
	CRM	22.6694	3.7724	0.8337
K=40	MaxEnt	24.4098	3.8068	0.8515
	CRM	24.5960	3.7182	0.8596
K=50	MaxEnt	25.6668	3.8068	0.8641
	CRM	25.9282	3.7479	0.8748

does worth the effort for the RL agent to spend more time to gather information to increase the chance of recommending the target item, thus receiving the reward.

4.4.2 Online User Study

We further evaluate our trained model with the online crowdsourcing experiments and present the quantitative results here. The ideal users would be those yelp users who have actually visited a number of restaurants and would like to chat with our agent to inform their current interest of a target. And our agent would make a recommendation at the end based on their historical interests as

well as the current session’s user intention. However, it is relatively difficult for us to find those real users. Instead, we come up with an experimental design to try to recover the ideal scenario as much as possible.

First, we randomly sample a target restaurant from the test dataset, containing a user id, a restaurant id, and the facets of this restaurant. Second, using the sampled user id, we retrieve a list of the restaurants that are visited by this user from the train set. This list is treated as the “historical” visiting information. Next, an MTurk is presented with the list of visited restaurants with their metadata. They are instructed to view each item carefully, in order to “learn the preference” of the sampled user. After this, the worker starts to chat with the conversational recommendation agent. The worker is presented with the facet values of the target restaurant on the side so that she can correctly answer the questions. However, she does not know which restaurant is the target. The agent may fail due to the error in the belief tracker module. When are recommended with a list of items, the worker needs to select up to three restaurants in the list. To motivate the worker to work carefully, she would receive a bonus if she successfully finds the target restaurant. There are no particular constraints, and workers can chat freely in natural language and end a session as they want.

The MaxEnt method based agent is used as a baseline since it is better than the others. Both of the two agents use the same belief tracker and recommender. Each worker needs to complete 8 dialogues. A worker interacts with our two agents in a random order, and each subject would not experiment with the same target restaurant more than once.

We collected 402 dialogues, with 201 dialogues for each method. The linear r_p recommendation reward is used. We set $C = 40$ and the stop threshold to 30. A dialogue is successful only if the worker finds the target restaurant from the

Table 4.8: Comparison of CRM and the MaxEnt method on R (Average Return), T (Average Turn), and S (Success Rate) in online user study.

Model	R	T	S
MaxEnt	1.8740	3.8457	31.84
CRM	4.0049	3.7114	36.82

recommendation list.

The results in Table 4.8 show that in general CRM is better, with a higher return of 4.0049. Note that the average return and the success rate are lower than the offline experiments, indicating that the real online settings are more complicated than offline settings. CRM achieves a higher success rate of 36.82%, showing that it can better assist real users to find the target items. Specifically, 74 out of 201 users successfully find their target items when using CRM, versus 64 out of 201 for the baseline. CRM has a shorter turn of 3.7114 compared to 3.8457 of the baseline. This is aligned with offline experiments because CRM is able to make a recommendation more aggressively.

We further explored the average ranking of the target item in the recommendation list, and that of the policy gradient model is 4.45, and that of the baseline is 3.83. The reason for this is because the MaxEnt model often has a shorter candidate list because it adds more constraints due to the longer dialogues. Table 4.9 shows the example dialogues collected in the user study. The MaxEnt method asks all slots before recommendation, while CRM often recommends quicker.

4.5 Summary and Contribution

This chapter describes the conversational recommendation framework, its implementation, and evaluation. Our framework has three components. The belief tracker computes a belief vector based on the dialogue utterances, describing the

Table 4.9: User study example dialogues of CRM and the MaxEnt methods. The facet values extracted by the belief tracker module are shown in the parentheses.

CRM	MaxEnt
<p><u>User:</u> I'm looking for mexican food <i>(category="Mexican")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.0 or above <i>(rating ≥ "4.0")</i> <u>Agent:</u> Which city are you in? <u>User:</u> Gilbert <i>(city="Gilbert")</i> <u>Agent:</u> Which price level would you like? <u>User:</u> medium <i>(price="medium")</i> <u>Agent:</u> <recommend, target ranked 1 of 8></p>	<p><u>User:</u> Mexican <i>(category="Mexican")</i> <u>Agent:</u> Which city are you in? <u>User:</u> Gilbert <i>(city="Gilbert")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.0 <i>(rating ≥ "4.0")</i> <u>Agent:</u> Which price level would you like? <u>User:</u> medium <i>(price="medium")</i> <u>Agent:</u> Which state are you in? <u>User:</u> AZ <i>(state="AZ")</i> <u>Agent:</u> <recommend, target ranked 1 of 5></p>
<p>Note: The RL agent asked for four slots and then make an recommendation, no need to ask for the rests.</p>	<p>Note: The rule agent collected all five slots before recommendation</p>
<p><u>User:</u> Find a restaurant in Henderson, NV <i>(state="NV", city="Henderson")</i> <u>Agent:</u> Which category do you want? <u>User:</u> Pizza <i>(category="Pizza")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.5 <i>(rating ≥ "4.5")</i> <u>Agent:</u> <recommend, target ranked 1 of 2></p>	<p><u>User:</u> Restaurant in Henderson NV <i>(state="NV", city="Henderson")</i> <u>Agent:</u> Which category do you want? <u>User:</u> Pizza <i>(category="Pizza")</i> <u>Agent:</u> Which rating range do you want? <u>User:</u> 4.5 or above <i>(rating ≥ "4.5")</i> <u>Agent:</u> Which price level would you like? <u>User:</u> medium <i>(price="medium")</i> <u>Agent:</u> <recommend, target ranked 1 of 1></p>
<p>Note: four slots are asked by the agent</p>	<p>Note: five slots are collected by the agent</p>

facet values constraints given by users. The recommender module takes the belief vector as additional features to predict a personalized item list for users. The reinforcement learning based dialogue manager decides when to ask facets and when to make a recommendation. We compared our proposed CRM model with the maximum entropy based baselines. On the Yelp dataset, the proposed method performs significantly better than the baselines because it aims at the maximization of the long term expected return. Instead of always asking a facet if there are still uncertainties about the facet values, the reinforcement learning agent learns that sometimes it can make a recommendation right away. We conduct both offline experiments and online user studies to validate the robustness and effectiveness of our proposed framework.

The major contributions of this work include the follows:

- Propose the way of modeling conversational recommendation problem: such a system should first chat with users to understand the user need and then make recommendations when the system is confident enough.
- Propose the framework and design different modules in the framework. The reinforcement learning based dialogue agent can decide when to ask facets and when to make a recommendation, using the context information predicted by the neural belief tracker, and recommend with the FM based recommender.
- Conduct offline experiments using simulated environments and test the trained model in online user studies to show that the proposed model can reach a better performance than baselines.

Chapter 5

Personalization in Conversational Recommender System

5.1 Overview

In this chapter, we explore how to better address the personalization problem in the framework to improve the dialogue performance. The recommendation in conversational dialogue is different from traditional scenarios, as users provide real-time, explicit feedback to the system. How to better utilize the user utterances to improve the recommendation performance is an open question. Figure 5.1 shows the proposed framework in this chapter. Again we have three components: a belief tracker, a recommender, and a dialogue manager. There are two major differences. First, we train a personalized probabilistic recommender to predict the probability of a user likes an item conditioned on the user past preferences, which is the personalized information, and the current in-session user interest. Second, we integrate the new recommender into the conversational recommendation framework by feeding the item probability distribution and the belief vector

to the reinforcement learning module to learn an improved policy. The details are introduced in the following sections.

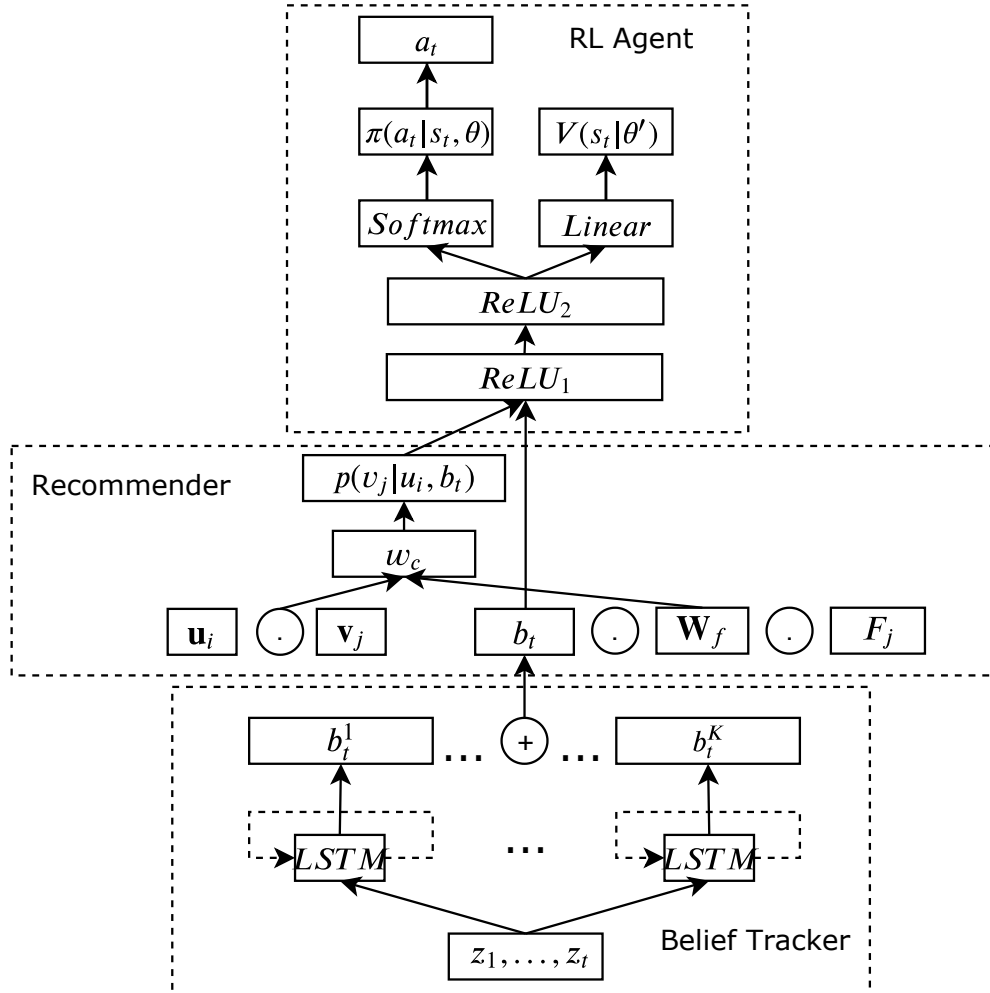


Figure 5.1: The proposed framework. The bottom part is the belief tracker, the middle part is the personalized probabilistic recommender, and the top part is the dialogue manager.

5.2 Methodologies

5.2.1 Belief Tracker

We use the same belief tracker as in the section 4.2.1, as shown in the bottom of Figure 5.1. Suppose there are K types of facets, for each facet $f_m (1 \leq m \leq K)$, the belief tracker predicts a probability distribution b^m over the possible facet values. It then concatenates the b^m into a belief vector. This vector contains the current belief of the facet values that have been addressed by the user in the dialogue. Let e_t denote the utterance of the dialogue turn t , we convert it to a k -gram ($k = 2$) sparse encoding z_t . For each facet f_m , we train an *LSTM* with the softmax activation layer to generate a probability distribution over the values of f_m :

$$b_t^m = LSTM(z_1, \dots, z_t) \quad (5.1)$$

Let $D = \sum_{m=1}^K |f_m|$ be the dimension of all the facet values, where $|f_m|$ is the number of values of the facet f_m . Thus the belief vector at dialogue turn t is the concatenation of the K facet value distributions: $b_t^T = (b_t^1 \oplus b_t^2 \dots \oplus b_t^K)^T \in \mathbb{R}^{D \times 1}$. This vector is then used as the input to the recommender and the dialogue manager, as will be introduced in the following sections.

5.2.2 Personalized Probabilistic Recommender

At each turn of the dialogue, given the current user and the dialogue utterances e_1, e_2, \dots, e_t , we want to recommend a ranking list of all the items. This ranking list keeps updating as the dialogue proceeds, and more constraints are being added. Gradually the target item is ranked to the top positions. It is preferable if the ranking list is on the whole items because the candidate set generation step of the recommender is not needed. Ideally, the target item can be ranked to the

first position in the list. Note that there might be multiple items meeting a user dialogue utterance constraints, and therefore might also be preferable to this user. For example, if a user likes a restaurant with the facet values of “(cuisine, Indian)” and “(price, medium)”, there are often many other restaurants meet these requirements. So it is, to some extent, reasonable to rank these restaurants in the top positions of the list. However, we do not have the ratings of these items. Therefore, in this work, we mainly evaluate the ranking of the real target item. Our results can be viewed as a lower bound of the quality of the system.

To achieve good recommendation results, we need to look at both the personalized user historical preferences towards items, as well as the user’s current interest. The dialogue utterances contain the real-time user’s need about the target item, which may deviate from the user’s past preference. In the meantime, the user’s historical preferences of items reflect the user’s personalized, general interests towards items. We argue that both signals are essential. Our intuitions are two folds. First, the facet constraints in the utterances express which kinds of items the user wants. As more restrictions are being added with the dialogue rolling, the target item would be gradually ranked to the top positions due to the facet matching. Second, if multiple items having similar facet values are ranked high, the personalized information can further differentiate the target item from its similar competitors.

Modeling the personalized user preference

Suppose there are M users $\{u_1, u_2, \dots, u_m\}$ and N items $\{v_1, v_2, \dots, v_n\}$. The users and items are mapped to a f dimensional latent embedding space [38], with the embedding describing the strength of a user or an item possesses towards a factor. The dot product of the user embedding $\mathbf{u} \in \mathbb{R}^\alpha$ and the item embedding

$\mathbf{v} \in \mathbb{R}^\alpha$ measures how close a user is to an item. Generally, it indicates how much a user like an item. Given the conversational recommendation dataset, we can train the user and item embeddings, which contain the personalized user preferences towards the items.

We pretrain the user-item embeddings with the negative sampling loss, to enable the embedding to contain both the closeness information and the ranking information, e.g., to rank the target item higher than the other items. Specifically, we use the top one probability loss function [9] [99] and an FM model. This loss function is designed to differentiate the target item (the positive item) from a set of negative items, and it is defined as

$$P(v_j) = \frac{\phi(s_{v_j})}{\sum_{j'} \phi(s_{v_{j'}})} \quad (5.2)$$

where $\phi(s_{v_j})$ is a monotonic positive increasing function about a scoring function s_{v_j} on item v_j . Here the rating predicted by the FM model is chosen as the scoring function, and the exponential function is used for ϕ , which converts $p(v_j)$ to the softmax function:

$$P(v_j) = \frac{\exp(s_{v_j})}{\sum_{j'} \exp(s_{v_{j'}})} \quad (5.3)$$

For each training instance, n negative samples are used for the pretrain of the user-item embeddings. At the time of training, we feed $n + 1$ training instances with the target item labeled 1 and the negative items labeled 0 to the model. The learning object is the cross-entropy error function. Then the trained embeddings are used to initialize the user-item embeddings in the personalized probabilistic recommender.

Modeling the current in session user interest

The belief tracker predicts a belief vector based on the dialogue history e_1, \dots, e_t . This vector contains the facet constraints of a user and is useful for ranking items. For example, the utterance “I want to find a comic movie” is strong evidence for ranking the comic movies higher than the other genre movies. We denote the 0 – 1 facet value vector of an item v_j as $F_j \in \mathbb{R}^{D \times 1}$, with the values of elements corresponding to the facet values of this item 1, and the rest elements value 0.

$$F_j = \mathbb{1}^D (\forall F_d = 1) \tag{5.4}$$

To utilize the facet information contained in the dialogue scripts, we compute the closeness of a belief vector b_t to F_j . The intuition is that an item that is “close” to the belief tracker is more likely to be the target item. Note that the belief tracker is not perfect and may contain errors. Specifically, the distance is defined as $b_t W_f F_j \in \mathbb{R}$, where $W_f \in \mathbb{R}^{D \times D}$ is a square parameter matrix. Note that W_f learns both the same facet value matching and the cross-facet value matching. For example, a movie starring “Will Smith” is very likely to be a comic movie. Thus, the cross facet matching also contributes to the dialogue item matching.

The personalized probabilistic recommender

Combining the above two types of information, the tuple $(\mathbf{u}_i \cdot \mathbf{v}_j, b_t W_f F_j)$ contains the user-item embedding matching score and the dialogue-item matching score, of a user u_i , an item v_j and a dialogue at turn t . Hence we define the

conditional probability of an item given a user and a dialogue belief vector as:

$$P(v_j|u_i, b_t) = \frac{\exp(w_c \cdot (\mathbf{u}_i \cdot \mathbf{v}_j, b_t W_f F_j))}{\sum_{j'} \exp(w_c \cdot (\mathbf{u}_i \cdot \mathbf{v}_{j'}, b_t W_f F_{j'}))} \quad (5.5)$$

where the denominator is summing over all items, and $w_c \in \mathbb{R}^2$ is a weighted summing parameter.

The direct optimization of this probability is computationally inefficient as we have to compute all the items in the denominator for each gradient update, and the number of items is usually large in recommendation datasets. Alternatively, we learn this conditional probability approximately by the negative sampling loss, which is introduced in [64]. The learning object is to distinguish the target item from a set of negative samples:

$$\log \sigma(w_c \cdot (\mathbf{u}_i \cdot \mathbf{v}_j, b_t W_f F_j)) + \sum_{j'} E_{j' \sim P(j')} \sigma(-w_c \cdot (\mathbf{u}_i \cdot \mathbf{v}_{j'}, b_t W_f F_{j'})) \quad (5.6)$$

In practice, we randomly sample $|j'| = 5$ items as the negative samples, following $P(j')$, which is the uniform distribution raised to the 3/4rd power. At test time, we use the trained model to rank all items and select the top ones to form a recommendation list for the user. The proposed recommender is shown in the middle part of Figure 5.1.

5.2.3 The Reinforcement Learning based Dialogue Agent

We train a reinforcement learning agent for the dialogue manager module of the framework. As mentioned earlier, the RL process is commonly modeled as an MDP, which contains the tuples of $\{s_t, a_t, r_t, \pi(a_t|s_t), \gamma\}$.

State: We use the dialogue belief vector b_t and the item conditional probability distribution, computed in section 5.2.2, to construct the dialogue state. We also

add the dialogue statistics s_{stat} vector as the additional features, which includes the number of dialogue turn so far, and the number of possible candidates computed by the belief vector. Thus the state is denoted as:

$$s_t = b_t \oplus P(v_1|u_i, b_t) \oplus \dots \oplus P(v_N|u_i, b_t) \oplus s_{stat} \quad (5.7)$$

Action: Similar to the previous chapter, we use two types of actions. One type is to decide to request more information and select a facet to ask the user based on the current understanding of the dialogue. The other is to decide to make a recommendation.

Reward: The reward is given by the environment, which is a simulated or real user. The agent gets a positive reward r_p if the target item in the top K positions of the recommendation list and a negative reward r_q if the dialogue fails, including cases that the user quit the dialogue, the target item is ranked too low, and the agent misunderstands the user. Again, we use a per dialogue turn r_c reward for preventing the dialogue from running too long.

Policy: We use the A3C model [66] to learn the policy. The vanilla policy gradient algorithm is a Monte Carlo method. It has the disadvantage of high variance and slow learning speed. By introducing a baseline to it, the Policy Gradient with Baseline approach remains unbiased, and the variance can be reduced. A normal choice of the baseline value would be a state value function $V(s_t|\theta')$.

$$\theta_{t+1} = \theta_t + \alpha (G_t - V(s_t|\theta')) \frac{\nabla \pi(a_t|s_t, \theta_t)}{\pi(a_t|s_t, \theta_t)} \quad (5.8)$$

However, this approach still suffers from the problems of high variance and slow learning speed. The one-step Actor-Critic methods remedy this by introducing bias into learning via bootstrapping. The bootstrapping refers to using the value

estimations of the subsequent states to update the current state’s value estimation. The introduced bias is beneficial as it reduces the variance, and therefore speed up the learning [109]. In this approach, a policy network, which is the Actor, picks an action to execute, and a value network, which is the critic, evaluates the decision. The one-step Actor-Critic method is an on-policy method.

The Asynchronous Advantage Actor-Critic (A3C) method by Mnih et al. [66] further improves the Actor-Critic by using the n-step return for the updating of the two networks. It uses multiple CPUs as the parallel learning processors, and update the global network asynchronously.

A3C is an off-policy biased model, with the benefits of speeding up the learning and reducing the variance. The learning object is to maximize the expected total Advantage starting from the beginning of the dialogue, with the Advantage defined as [66]:

$$A(s_t, a_t|\theta, \theta') = \sum_{i=0}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k}|\theta') - V(s_t|\theta') \quad (5.9)$$

This method maintains both a policy network $\pi(a_t|s_t, \theta)$ and a value network $V(s_t|\theta')$, with θ and θ' the parameters respectively. We use two *ReLU* layers with a softmax layer at the top as the policy network, and a linear layer at the top as the value network, as shown in the top part of Figure 5.1. The update performed is:

$$\begin{aligned} \theta_g &= \theta_g + \nabla_{\theta} A(s_t, a_t|\theta, \theta') \log \pi(a_t|s_t, \theta) \\ \theta'_g &= \theta'_g + \nabla_{\theta'} (G_t - V(s_t|\theta'))^2 \end{aligned} \quad (5.10)$$

where the θ_g and θ'_g are parameters in the global network and are updated asynchronously by thread-specific parameters θ and θ' . The Advantage is computed from the current time step t to the end of the episode. The learning algorithm is shown in Algorithm 3. Our readers may refer to [66] [108] for more details.

Algorithm 3 The learning process of the A3C agent.

- 1: Start with \mathcal{M} epochs, \mathcal{N} training data
 - 2: Initialize the global policy network θ_g and the global value network θ'_g
 - 3: **for** epoch = 1, \mathcal{M} **do**
 - 4: **for** i = 1, \mathcal{N} **do**
 - 5: Synchronize parameters in the thread $\theta = \theta_g, \theta' = \theta'_g$
 - 6: $t = 0$, set episode set $\mathcal{E} = \{\}$
 - 7: Sample a dialogue with starting utterance e_t , user u_i
 - 8: **while** not dialogue ends **do**
 - 9: Apply belief tracker to $e_{0:t}$ to get b_t
 - 10: Apply the recommender to get $P(v_j|u_i, b_t)$
 - 11: $s_t = b_t \oplus P(v_1|u_i, b_t) \dots \oplus s_{stat}$, sample a_t from the policy network
and get the state value V_t from the value network.
 - 12: Get r_t and user response from the user simulator
 - 13: Add s_t, a_t, r_t, V_t to \mathcal{E}
 - 14: **If** a_t is recommend, **then** dialogue ends
 - 15: $t = t + 1$
 - 16: $G_t = 0$
 - 17: **for** each step in \mathcal{E} **do**
 - 18: $G_t = r_t + \gamma G_t$
 - 19: Asynchronously update θ_g with $\nabla_{\theta} \pi(a_t|s_t)(G_t - V(s_t|\theta'))$
 - 20: Asynchronously update θ'_g with $\nabla_{\theta'} (G_t - V(s_t|\theta'))^2$
-

5.3 Experimental Settings

5.3.1 User Simulation

We use simulated users to train the conversational recommender agent. Specifically, at turn t , the probability of a user continue to the next turn is p_c^t . This probability decreases exponentially as the turn t increases. The higher the p_c is, the more patient the simulated user is. Similar as before, the user gives a positive reward r_p if it finds the target item in the top K positions in the list. Otherwise, a negative reward r_q is returned. The user generates a small negative reward r_c each turn if the dialogue continues.

5.3.2 Evaluation Measures and Baseline for the Personalized Probability Recommender

Evaluation measure

Since there is always only one target item, we hope that the target item is ranked as high as possible. To this end, the Mean Reciprocal Rank (MRR) measure on the top 100 positions of the target item is used for evaluation. Let N denotes the set of target items on the evaluation dataset, $MRR@k$ is defined as:

$$MRR@k = \frac{1}{|N|} \sum_i^{|N|} \frac{1}{\text{ranking}_i} \quad (5.11)$$

Baselines

We use a number of baselines, including the start-of-the-art recommendation algorithms that include the pointwise and pairwise methods.

(1) Factorization Machine [81], a pointwise recommender. It takes the user-item one hot vector to predict the rating score of a user-item pair.

(2) Bayesian Personalized Ranking (BPR) [82], which is a pairwise model. This recommender considers the pairwise ranking of the target item and a random drawn negative sample, aiming to maximize the posterior estimate from a Bayesian point of view. It optimizes:

$$BPR_{OPT} = \sum_{(u,i,j) \in D_s} \ln \sigma(\hat{x}_{uij}) - \lambda \|\Theta\|^2 \quad (5.12)$$

And for an MF-BPR,

$$\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj} \quad (5.13)$$

(3) ListRank-MF [99]: this is a listwise method based on the matrix factorization approach. It trains a model to learn the object:

$$ListRank_{OPT} = \operatorname{argmin} \frac{1}{2} \sum_{i=1}^M \sum_{j=1}^N I_{ij} (R_{ij} - g(U_i^T V_j))^2 + \frac{\lambda_U}{2} \|U\|^2 + \frac{\lambda_V}{2} \|V\|^2 \quad (5.14)$$

where the R_{ij} is the original rating, and $g(U_i^T V_j)$ is the learned prediction with the user and item represented by the low-rank vector U_i and V_j respectively. The top-one probability [9] is used to learn the parameters.

5.3.3 Evaluation Measures and Baselines for the Dialogue Agent

Evaluation measures

Similarly, we evaluate the framework on three dialogue measures: the Average Return, the Average Length, and the Success Rate, as introduced in Section 4.3.2. Note that the success rate is also viewed as the Conversion Rate in the recommendation field.

Baselines

We use the MaxEnt and the CRM model in the previous chapter as baselines. The MaxEnt policy always selects the facet that has the maximum entropy on the values to ask [18]. It recommends when the max entropy of all facets is below a threshold. We conduct a grid search to identify the best performance threshold is 3.5. As introduced before, the CRM model contains a belief vector, an FM recommender, and a policy gradient dialogue manager.

Training parameters

We use only the items that have ratings ≥ 4 for training because these user-item pairs are denoted as “high quality” user-item preferences. We randomly split the dataset into a train, dev, and test set with an 80%, 10%, 10% percentage. Models are trained and evaluated on the train and dev set. Experiments are evaluated on the test set. There are 15 facets on the MovieLens1M dataset and 23 facets on the Yelp dataset. The belief tracker hidden size is 64. The user-item embedding size is 64, and the negative sampling loss is computed with 5 negative samples. For the RL parameters, we use the RMSProp optimizer with a learning rate 0.001, $\gamma = 0.99$, $r_p = 30$, $r_q = -10$, $r_c = -1$, $K = 30$. The continue probability of the user is $p_c = 0.8$.

5.4 Experiments Results

5.4.1 The Personalized Probabilistic Recommendation

First, we evaluate the performance of the personalized probabilistic recommender on the collected conversational recommendation dialogue dataset. The recommender introduced above generates a recommendation list of all the items,

Table 5.1: Comparison of the Non-personalized Recommender (NPR) and the Personalized Recommender (PR) with baselines. Evaluated on the MRR@100 of the target item, when different number of facets are known. The top part is on the MovieLens1M dataset, and the bottom part is on the Yelp dataset. Results in bold are significantly better than baselines at 0.01 level.

MovieLens1M	1	2	3	4	5	6	7
FM	0.0120						
BPR-MF	0.0317						
ListRank-MF	0.0191						
NPR	0.0969	0.2552	0.4180	0.5614	0.6804	0.7847	0.8565
PR	0.1133	0.2796	0.4438	0.5786	0.6916	0.7846	0.8528
Yelp	1	2	3	4	5	6	7
FM	0.0011						
BPR-MF	0.0171						
ListRank-MF	0.0034						
NPR	0.0249	0.0791	0.1516	0.2501	0.3654	0.4843	0.5943
PR	0.0610	0.1322	0.2179	0.3136	0.4191	0.5237	0.6159

at each turn in the dialogue, based on both the past user preferences and the current in-session user interest. As the dialogue continues, more facet constraints are being added. We want to evaluate the quality of the recommendation as the number of known facets increases.

We explore two settings of our model: one with dialogue-item matching information only, and we denote it as the Non-Personalized Recommender (NPR); the other uses both personalized user-item matching information and the dialogue-item matching information, which is denoted Personalized Recommender (PR).

Results: As shown in Table 5.1. The top part is on the MovieLens1M dataset, and the bottom part is on the Yelp dataset. First, the two proposed models perform significantly better than the baselines on both datasets. Second, on MovieLens1M dataset, PR reaches a higher MRR when fewer facets are known, for example, when $\#known\ facets \leq 5$. Because when less information is known, the user-item matching information help with differentiating the target item from the

other items that have similar facet values. However, when $\#known\ facets \geq 6$, NPR performs better because when more facet values are known, the dialogue-item matching information alone can make a good recommendation. Since the dialogue usually will not run to a large number of turns, this indicates that PR is a beneficial approach. On the Yelp dataset, PR is constantly better than NPR. Overall, the experiment results illustrate that our approach of modeling the conversational recommendation problem is effective, and specifically, the personalized information is useful for making recommendations in the dialogue.

5.4.2 The Personalized Conversational Recommender System

In this part, we evaluate the dialogue performance by integrating the personalized probabilistic recommender into the conversational recommender framework. At each turn, our trained RL agent interacts with a simulated user and decides whether to select a facet to ask the user or to make a recommendation. If it decides to recommend, it presents the item list computed by the personalized probabilistic recommender.

Offline experimental results

The major results are shown in Table 5.2, with MovieLens1M dataset results on the top, and Yelp dataset results at the bottom. We use the PA to represent the Personalized RL Agent and NPA to represent the Non-Personalized RL agent. First, our two probabilistic recommender based dialogue agents outperform both the maximum entropy rule-based agent and the CRM agent, by reaching a higher average return, a shorter average dialogue length, and a higher success rate. This is because we use the dialogue-facet matching (for both PA and NPA) and per-

Table 5.2: The conversational recommendation dialogue performance of the Non Personalized Agent (NPA) and the Personalized Agent (PA) with the simulated user on the MovieLens1M Dialogue Dataset (the upper part) and on the Yelp Dialogue Dataset (the lower part). Results in bold are significantly better than baselines at 0.01 level.

MovieLens1M	Avg. Return	Avg. Turn	Success Rate
Max Entropy	2.5293	3.2104	0.3827
CRM	5.2727	2.8032	0.4837
NPA	10.2151	2.7417	0.6187
PA	11.0647	2.5961	0.6393
Yelp	Avg. Return	Avg. Turn	Success Rate
Max Entropy	-1.9505	3.1140	0.2970
CRM	-0.3199	3.1609	0.3538
NPA	7.0484	2.8936	0.5493
PA	8.9326	2.7326	0.5988

sonalized user-item matching information (for PA) in the recommender module. Second, PA is significantly better than NPA in all three measures, illustrating that the personalization information can further improve the conversational recommendation performance. The experiment results are consistent on the two datasets.

Two exemplary dialogues are shown in Table 5.3. The dialogue length is the shorter, the better, and the item is ranked the higher, the better. The upper part shows PA and NPA interacts with a simulated user on an instance from the MovieLens1M dataset. For example, we observe that the user starts with an utterance contains the genre preference, and the agent responds by asking more facets and makes a recommendation when enough values information are collected. We can see that PA and NPA make the recommendation in the second and the third round, respectively. Also, the target item is ranked to the 3rd position in PA’s recommendation list versus the 28th position in NPA’s recommendation list. The lower part shows two example dialogues from the Yelp Dataset. The two

Table 5.3: Comparison of example dialogues between PA and the NPA on the MovieLens1M dataset (the top row) and the Yelp dataset (the bottom row).

Personalized Agent	Non Personalized Agent
<p><u>User</u>: my genre of choice is Drama <i>(genre="Drama")</i></p> <p><u>Agent</u>: which meta critic rating range would you like?</p> <p><u>User</u>: the meta critic rating range should be 70-80 <i>(rating_range="70-80")</i></p> <p><u>Agent</u>: recommend you with the following items (item ranked 3 of 30)</p> <p><u>User</u>: thank you</p>	<p><u>User</u>: my genre of choice is Drama <i>(genre="Drama")</i></p> <p><u>Agent</u>: which rating range would you like?</p> <p><u>User</u>: a movie with a rating of 7.0 to 8.0 <i>(rating_range="7.0-8.0")</i></p> <p><u>Agent</u>: how many reviews would you like the movie to have?</p> <p><u>User</u>: i would like to see a movie with review 0-50000 <i>(num_review="0-50000")</i></p> <p><u>Agent</u>: recommend you with the following items (item ranked 28 of 30)</p> <p><u>User</u>: thank you</p>
Personalized Agent	Non Personalized Agent
<p><u>User</u>: find me a medium price Chinese restaurant <i>(category="Chinese", price_range="medium")</i></p> <p><u>Agent</u>: which rating do you want?</p> <p><u>User</u>: i want a restaurant with 3.5 rating please <i>(rating="3.5")</i></p> <p><u>Agent</u>: recommend you with the following items (item ranked 1 of 30)</p> <p><u>User</u>: thank you</p>	<p><u>User</u>: find me a medium price Chinese restaurant <i>(category="Chinese", price_range="medium")</i></p> <p><u>Agent</u>: which city are you in?</p> <p><u>User</u>: around Las Vegas, NV <i>(city="Las Vegas, NV")</i></p> <p><u>Agent</u>: recommend you with the following items (item ranked 3 of 30)</p> <p><u>User</u>: thank you</p>

agents recommend at the same turn, with the target item ranked higher by PA than NPA.

The detailed analysis by the dialogue turn

We dig deeper to compare the dialogue performances at different dialogue turns of the two proposed models. Results are shown in Figure 5.2. First, on both datasets, both PA and NPA have many more short turn (turn length 2 & 3) dialogues than the long turn (turn length 4 or more) dialogues. Second, PA has more short turn dialogues than NPA, which explains the reason PA has a shorter aver-

age dialogue length. Third, PA generally has a higher average return and success rate on the short dialogues than NPA. This illustrates that PA tends to make a recommendation more quickly because it can reach recommendation performance when fewer facets are known. While NPA is better at longer dialogues, in which case more constraints are added by the dialogue utterances. This illustrates that when fewer facets are known, the personalization information plays an important role. However, when more facets values become available, the facet values alone can provide strong enough information for finding the target item, in which case NPA can have a better performance.

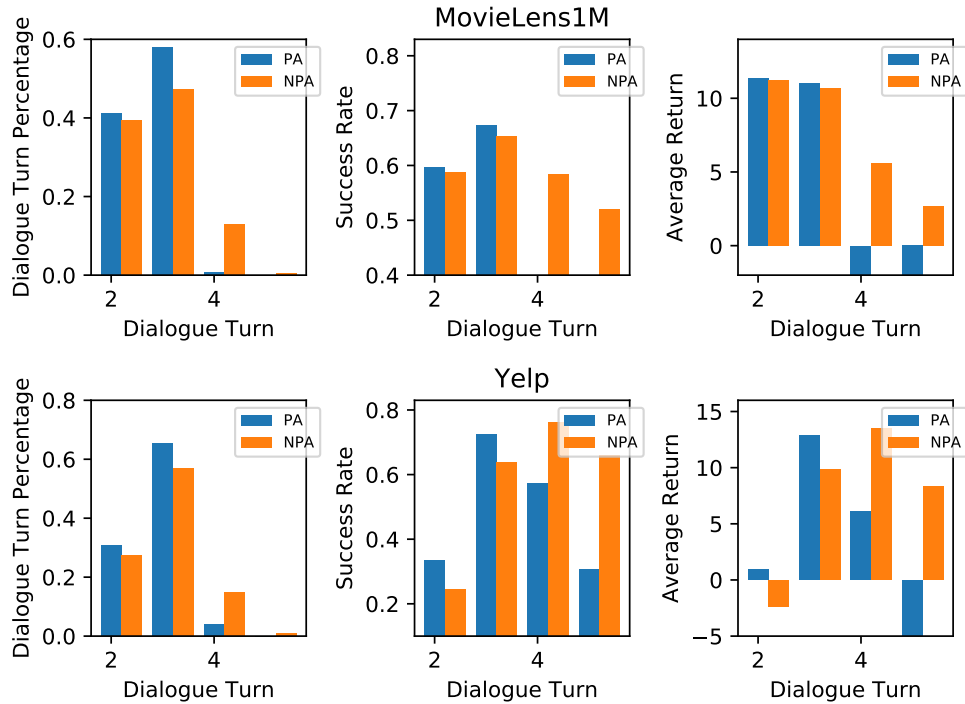


Figure 5.2: Comparing the percentage of dialogue turns, the success rate and the average return of the PA and NPA at different number of dialogue turns, on the MovieLens1M dataset (the upper row) and the Yelp dataset (the lower row).

Table 5.4: Comparing the conversational recommendation performance of the PA and NPA interacting with simulated users having different continue probability p_c .

	MovieLens1M			Yelp		
$p_c = 0.85$	R	T	S	R	T	S
NPA	12.4057	2.9982	0.6769	9.3126	3.1524	0.6145
PA	12.5453	2.8703	0.6844	10.5880	2.9289	0.6446
$p_c = 0.75$	R	T	S	R	T	S
NPA	8.8174	2.5801	0.5799	2.1888	2.4700	0.4092
PA	9.8127	2.4937	0.6075	7.8568	2.6672	0.5647
$p_c = 0.70$	R	T	S	R	T	S
NPA	8.4335	2.4953	0.5667	-3.0171	2.0001	0.2399
PA	9.1623	2.4491	0.5821	5.5274	2.4400	0.5075

The effects of different user continue probability p_c

Since our model is trained with interactions with the simulated user, we further evaluate the performance by varying the behavior of the simulated user, which is the continuing probability p_c at each round. We test it at different levels, from 0.70 to 0.85 and the results are shown in Table 5.4. Note that p_c decays exponentially as the dialogue grows.

We observe that in general, on both datasets, as p_c decrease, all the measures also decrease, because the user has a higher probability of leaving the conversation. Specifically, on the MovieLens1M dataset, PA is significantly better than NPA, with one exception when $p_c = 0.85$, the improvement on return is not significant. This is because MovieLens1M dataset is denser than the Yelp dataset. And the dialogue-item matching information alone can be reasonably effective. On the Yelp dataset, the PA is significantly better than NPA in the average return. When p_c is small, i.e., $p_c \leq 0.75$, the NPA agent performed poorly. This is also because the Yelp dataset is sparser than the MovieLens1M dataset; thus, the personalized user-item historical preference information has a big impact.

The advantage of A3C over policy gradient

We explore the advantage of using the A3C method. Specifically, we compare the Average Return gained using the vanilla policy gradient method and the A3C method, on both datasets. Results are shown in Figure 5.3, which shows the Average Return evaluated on the dev dataset against the training epochs. We observe that the A3C methods can converge to a higher Average Return, for both PA and NPA, than the policy gradient methods. The A3C methods can escape from the local minimum. For example, in the lower-left figure, the pg agents (the orange line) converge to a sub-optimal average return, while the A3C agents (the blue line) jumps to a higher return given more training time.

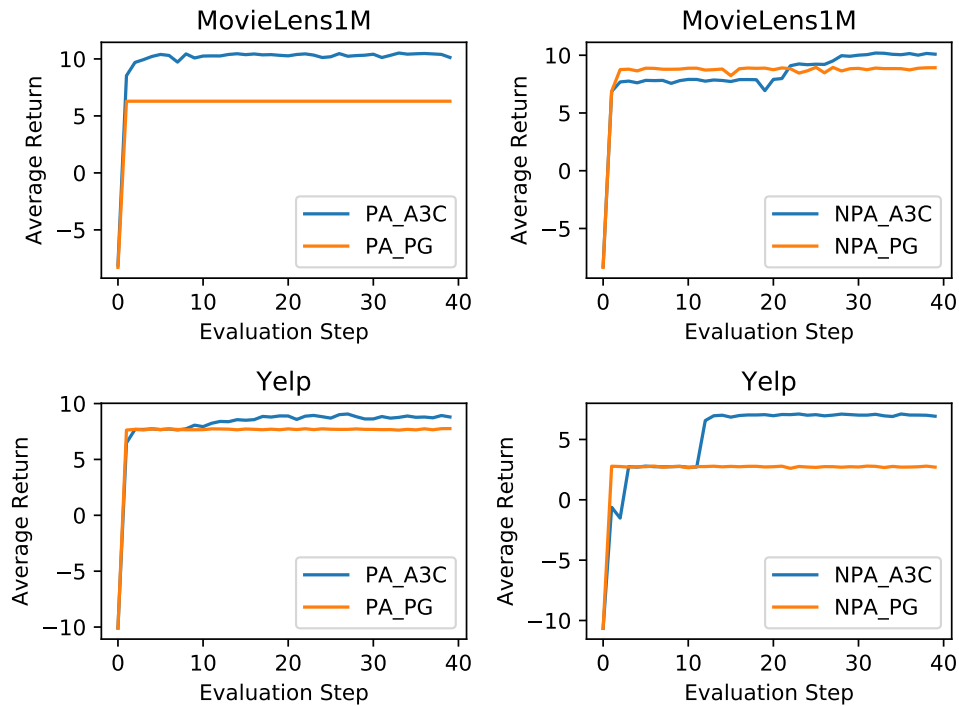


Figure 5.3: The Average Return obtained by using the Policy Gradient method and the A3C method. The x-axis is the total number of evaluation steps. The y-axis shows the Average Return evaluated on the whole dev dataset.

Table 5.5: The comparison of NPA and PA in online user study on the MovieLens1M dataset.

MovieLens1M	R	T	S
NPA	-2.8559	3.5339	25.69
PA	0.1192	3.3119	31.19

Table 5.6: The comparison of NPA and PA in online user study on the Yelp dataset.

Yelp	R	T	S
NPA	-3.764	3.3008	22.32
PA	0.111	3.0170	33.33

5.4.3 User Study

In order to evaluate our trained agents against real users, we conducted online user studies. We use the similar settings as in section 4.4.2. However, we have more facets in this user study. Again we first sample a user-item pair, then show the preferred historical items of the user to a crowdsourcing worker. After the worker has learned the sampled user’s preference, she starts to converse with our trained models. We show the target item’s facets to the worker, and let her chat with our agent to get recommendations. She must select 3 items in the list. The dialogue is deemed as a success if the worker successfully finds the target item. We collect 109 dialogues on the MovieLens1M dataset and 117 dialogues on the Yelp dataset, for both agents. Results are shown in Table 5.5 and Table 5.6. The results are consistent with the offline experiments, in that PA has a higher average return, shorter dialogue length, and higher success rate. Table 5.7 shows the comparison of PA and NPA on the collected user study dialogues.

Table 5.7: User study example dialogues of PA and NPA on the MovieLens1M dataset (the top row) and the Yelp dataset (the bottom row).

Personalized Agent	Non Personalized Agent
<p><u>User:</u> i like morgan freeman movie <i>(actor="Morgan Freeman")</i> <u>Agent:</u> which genre would you like? <u>User:</u> drama <i>(genre="Drama")</i> <u>Agent:</u> recommend you with the following items (item ranked 3 of 30) <u>User:</u> thank you</p>	<p><u>User:</u> okay <i>(no facets detected)</i> <u>Agent:</u> which genre would you like? <u>User:</u> Drama <i>(genre="Drama")</i> <u>Agent:</u> which rating range would you like? <u>User:</u> 7.0-8.0 <i>(num_review="dontknow")</i> <u>Agent:</u> how many reviews would you like <u>User:</u> 50000 to 100000 <i>(num_revie="50000-100000")</i> the movie to have? <u>Agent:</u> recommend you with the following items (item ranked 17 of 30) <u>User:</u> thank you</p>
Personalized Agent	Non Personalized Agent
<p><u>User:</u> Soul Food <i>(category="Sandwiches")</i> <u>Agent:</u> which rating do you want? <u>User:</u> 3.5 <i>(category="Sandwiches", rating="3.5")</i> <u>Agent:</u> recommend you with the following items (item ranked 1 of 30) <u>User:</u> thank you</p>	<p><u>User:</u> casual <i>(no facets found)</i> <u>Agent:</u> which category do you want? <u>User:</u> Soul Food <i>(category="Donuts")</i> <u>Agent:</u> which city are you in? <u>User:</u> Phoenix <i>(category="Soul Food", city="Phoenix")</i> <u>Agent:</u> recommend you with the following items (item ranked 2 of 30) <u>User:</u> thank you</p>

5.5 Summary and Contribution

In this chapter, we propose the personalized probabilistic recommender to predict the conditional probability of an item given a user and a dialogue context. This approach captures the dialogue-item matching information of the dialogue scripts and personalized user-item preference information. The facet value constraints in the dialogue utterances can effectively narrow down a subset of items with the target item ranked high. At the same time, the user past preference information can further differentiate the target item from its peers with similar facet values. The joint modeling of the two types of information leads to a recommendation module that is suitable for the conversation dialogue scenario. We compare the new recommender to the start-of-the-art models and show that it has better performance. Then we integrate the new module into the conversational recommendation framework and perform both offline and online experiments to demonstrate they can improve the dialogue performance.

The major contribution of this work includes the following:

- Propose the personalized probabilistic recommender to model the past user preferences in the historical user-item information and the user current session preferences in the dialogue utterances.
- Integrate the recommender into the conversational recommendation framework. Conduct experiments and show that the new recommender module improves dialogue performance, and specifically, the personalization information is beneficial in both offline and online experiments.

Chapter 6

Conclusion and Future Work

In this chapter, we conclude this dissertation by listing the contributions, discussing the limitations, and pointing out the possible future work directions.

6.1 Contribution

First, we contribute the conversational recommendation datasets in Chapter 3. We develop datasets based on two popular recommendation datasets. Our datasets have the conversational dialogues and the user-item preference information. We argue that a conversational recommendation agent should gradually elicit user preferences of items and then make recommendations accordingly. Thus we manage to collect dialogues with the utterances discussing the facet values of the target items. The dialogues in the datasets are collected with crowdsourcing experiments, and the quality of utterances are validated. The final simulated dialogues have rich facet value information, which supports the facet-based conversation, as well as rich user-item rating information, which supports the training of recommendation modules. Hopefully, the datasets can facilitate future researches on how to make recommendations in dialogue systems.

Second, we propose the conversational recommendation framework in Chapter 4. The framework contains a belief tracker, a recommender, and a dialogue manager. The three components work together to function as a recommendation oriented dialogue agent, that can decide whether to request more facet values from users or to make a recommendation when enough information has been collected. The proposed model aims at the maximizing of the long-term expected total rewards. Instead of always asking for more facets when there are uncertainties, our framework learns that sometimes it should make recommendations right away. We train the dialogue manager with different simulated users. Experiments with both simulated user and real users illustrate the effectiveness of the proposed framework.

Third, we propose a personalized probabilistic recommender to extend the framework. Recommending in the dialogue environment is different from traditional scenarios as there is real-time user feedback involved, and the dialogue agent can explicitly request information which is deemed as necessary from the user. The model can adequately consider the information contained in dialogue utterances as well as the personalized user preferences contained in the past. We integrate this recommender into the conversational recommendation framework and conduct extensive experiments. Results show that the personalized user preference is essential information in further improving both the recommendation and the dialogue performances.

6.2 The Limitations and Future Work

Our research in this dissertation is only one step towards integrating the dialogue system and recommender systems into a conversational recommender system. There is still much work to do in this direction.

6.2.1 Limitation of the Collected Conversational Recommendation Dataset

There are limitations of the datasets we introduced in Chapter 3. First, we use the simulation and the crowdsourcing methods to collect the datasets. However, compared to real human-human or human-machine dialogues, our dialogues are still not natural enough. A future improvement might be to use dialogues collected from a real-world conversational recommender system or real human-human conversation to build models. Alternatively, we can design crowdsourcing experiments to let two workers chat in the scenario of conversation recommendation. One worker plays the role of a virtual sales agent, and the other worker plays the role of a customer. However, how to design the right mechanism to provide incentives for crowdsourcing workers to behave like real sales/recommendation agents and buyers might be an important issue to consider. Second, the dialogues we collected are used to train the NLU module. The NLU model trained does not have much ability to recognize discourse information such as coreference resolution. Besides, if we want to use dialogues to train dialogue policy, we need data collected from a dialogue system with real users or better-simulated users to use this system to find their preferred target items, so that the datasets have more realistic recommendation related decision making information in them. Collaboration with industry or owning a real conversational recommender system with real users would be very helpful.

6.2.2 Joint Learning the Recommender and the Dialogue Policy

The recommender module in the framework is pre-trained in our framework. In the future, we should improve this by training the recommender and the dialogue manager jointly, for example, to allow the reward signals to update both the dialogue policy and the recommendation algorithm. Similar to the discussion in 6.2.1, we need datasets collected from real dialogue systems recommending items to real users. The dialogue agent needs to learn from sequences of dialogue decisions leading to both successful and unsuccessful recommendations, as well as which items to present in the recommendation list to gain more rewards. Moreover, we only consider the ranking of the real target item, and the other items are treated as “irrelevant” items because we do not have the labels of them. However, this is too harsh. The other items can be very similar to the target item and may also be preferable to users. Therefore, our experiment results can be interpreted as a lower bound of the real system. Datasets from the real conversational recommendation systems can overcome this problem, because the choosing of an item by a real user represents that, given the dialogue context, this user prefer this item to all the other items, and this real user behavior can provide us with real positive and negative user feedback, which can guide the joint learning of both the dialogue policy and the recommendation model.

6.2.3 Enabling More User Actions and Machine Actions

As shown in Chapter 4 and 5, the primary user actions in our dialogues include a user informs the dialogue agent facet value constraints or expresses one does not know the value, and the major machine actions are to request more facets from users and to make a recommendation for a user. There are more dialogue actions

need to be added to the dialogue in the future. For example, a user can request facet values from the dialogue agent so that both the dialogue and the user can take the initiative in the conversation. Moreover, the dialogue agent should be able to provide candidate facet values when requesting them from users, and the candidate suggestion can be learned from the recommendation dialogues as well. After issuing recommendations to a user, the user can further, for example, negate a facet value or update an informed value based on the recommended items, and the dialogue agent should be able to recommend new sets of items based on user feedback. This process enables the user and the dialogue agent to negotiate so that the recommended results gradually drifting towards the real user preference. There are more decision makings need to be learned in the session, for example, how much more information to collect before making the next recommendation, and how to consider the past sequence of user preferences in the current session to generate new recommendations. Finer simulation strategies are needed, or datasets from real industrial systems are needed to enable the research in these directions.

6.2.4 Integrating More Dialogue Functionalities

There is still a big gap between a real salesperson and a conversational recommendation agent. The future conversational recommender system should be a systematic integration of all three types of dialogue systems: the task-oriented dialogue system, the open domain dialogue system, and the question answering system. For example, a user can ask general questions towards one category of products to gain the knowledge needed for finding items. One can also ask specific questions about the recommended item. Moreover, it is favorable if the dialogue agent can respond naturally to chitchat of users at any turn in the dialogue,

which makes the dialogue agent behaves more like a real human agent. The above scenarios may interleave with each other in the whole dialogue process, and the dialogue agent should be able to track the states of multiple intentions correctly. Fulfilling any of the above cases can generate important gains, which should be adequately modeled to guide the training of the dialogue agent. Each of the above functions can be an interesting research direction to explore.

Bibliography

- [1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering*, (6):734–749, 2005.
- [2] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W Bruce Croft. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 645–654. ACM, 2017.
- [3] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.
- [4] Daniel Billsus and Michael J Pazzani. User modeling for adaptive news access. *User modeling and user-adapted interaction*, 10(2-3):147–180, 2000.
- [5] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. *arXiv preprint arXiv:1605.07683*, 2016.
- [6] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*, 2018.
- [7] Robin D Burke, Kristian J Hammond, and Benjamin C Young. Knowledge-based navigation of complex information spaces. In *Proceedings of the national conference on artificial intelligence*, volume 462, page 468, 1996.
- [8] Erion Çano and Maurizio Morisio. Hybrid recommender systems: A systematic literature review. *Intelligent Data Analysis*, 21(6):1487–1524, 2017.
- [9] 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 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

- [10] Oscar Celma. Music recommendation. In *Music recommendation and discovery*, pages 43–85. Springer, 2010.
- [11] Hongshen Chen, Xiaorui Liu, Dawei Yin, and Jiliang Tang. A survey on dialogue systems: Recent advances and new frontiers. *ACM SIGKDD Explorations Newsletter*, 19(2):25–35, 2017.
- [12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishikesh Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10. ACM, 2016.
- [13] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824. ACM, 2016.
- [14] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198. ACM, 2016.
- [15] Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94. ACM, 2008.
- [16] Luis M De Campos, Juan M Fernández-Luna, Juan F Huete, and Miguel A Rueda-Morales. Combining content-based and collaborative recommendations: A hybrid approach based on bayesian networks. *International Journal of Approximate Reasoning*, 51(7):785–799, 2010.
- [17] Mukund Deshpande and George Karypis. Item-based top-n recommendation algorithms. *ACM Transactions on Information Systems (TOIS)*, 22(1):143–177, 2004.
- [18] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 484–495, 2017.
- [19] Jesse Dodge, Andreea Gane, Xiang Zhang, Antoine Bordes, Sumit Chopra, Alexander Miller, Arthur Szlam, and Jason Weston. Evaluating prerequisite qualities for learning end-to-end dialog systems. *arXiv preprint arXiv:1511.06931*, 2015.

- [20] Enrique Estellés-Arolas and Fernando González-Ladrón-De-Guevara. Towards an integrated crowdsourcing definition. *Journal of Information science*, 38(2):189–200, 2012.
- [21] Jianfeng Gao, Michel Galley, Lihong Li, et al. Neural approaches to conversational ai. *Foundations and Trends® in Information Retrieval*, 13(2-3):127–298, 2019.
- [22] Marjan Ghazvininejad, Chris Brockett, Ming-Wei Chang, Bill Dolan, Jianfeng Gao, Wen-tau Yih, and Michel Galley. A knowledge-grounded neural conversation model. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [23] Hui Feng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. Deepfm: a factorization-machine based neural network for ctr prediction. *arXiv preprint arXiv:1703.04247*, 2017.
- [24] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):19, 2016.
- [25] Jianming He and Wesley W Chu. A social network-based recommender system (snrs). In *Data mining for social network data*, pages 47–74. Springer, 2010.
- [26] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee, 2016.
- [27] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, pages 173–182. International World Wide Web Conferences Steering Committee, 2017.
- [28] Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990*, 1990.
- [29] Matthew Henderson, Blaise Thomson, and Jason D Williams. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, 2014.

- [30] Matthew Henderson, Blaise Thomson, and Steve Young. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop (SLT)*, pages 360–365. IEEE, 2014.
- [31] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
- [32] Stefan Hillmann, Stefan Hillmann, and Baumann. *Simulation-Based Usability Evaluation of Spoken and Multimodal Dialogue Systems*. Springer, 2018.
- [33] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.
- [34] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using click-through data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.
- [35] Seokhwan Kim, Luis Fernando D’Haro, Rafael E Banchs, Jason D Williams, Matthew Henderson, and Koichiro Yoshino. The fifth dialog state tracking challenge. In *2016 IEEE Spoken Language Technology Workshop (SLT)*, pages 511–517. IEEE, 2016.
- [36] Jonathan Koren, Yi Zhang, and Xue Liu. Personalized interactive faceted search. In *Proceedings of the 17th international conference on World Wide Web*, pages 477–486. ACM, 2008.
- [37] 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*, pages 426–434. ACM, 2008.
- [38] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, (8):30–37, 2009.
- [39] Xuan Nhat Lam, Thuc Vu, Trong Duc Le, and Anh Duc Duong. Addressing cold-start problem in recommendation systems. In *Proceedings of the 2nd international conference on Ubiquitous information management and communication*, pages 208–211. ACM, 2008.
- [40] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436, 2015.

- [41] Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.
- [42] Cheong-Jae Lee, Sang-Keun Jung, Kyung-Duk Kim, Dong-Hyeon Lee, and Gary Geun-Bae Lee. Recent approaches to dialog management for spoken dialog systems. *Journal of Computing Science and Engineering*, 4(1):1–22, 2010.
- [43] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*, 2015.
- [44] Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*, 2016.
- [45] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.
- [46] Peng Li, Wei Li, Zhengyan He, Xuguang Wang, Ying Cao, Jie Zhou, and Wei Xu. Dataset and neural recurrent sequence labeling model for open-domain factoid question answering. *arXiv preprint arXiv:1607.06275*, 2016.
- [47] Xiujun Li, Yun-Nung Chen, Lihong Li, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. *arXiv preprint arXiv:1703.01008*, 2017.
- [48] Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. A user simulator for task-completion dialogues. *arXiv preprint arXiv:1612.05688*, 2016.
- [49] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.
- [50] Long-Ji Lin. Reinforcement learning for robots using neural networks. Technical report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.
- [51] Pierre Lison and Casey Kennington. Opendial: A toolkit for developing spoken dialogue systems with probabilistic rules. *Proceedings of ACL-2016 System Demonstrations*, pages 67–72, 2016.

- [52] Jiahui Liu, Peter Dolan, and Elin Rønby Pedersen. Personalized news recommendation based on click behavior. In *Proceedings of the 15th international conference on Intelligent user interfaces*, pages 31–40. ACM, 2010.
- [53] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*, pages 73–105. Springer, 2011.
- [54] Ryan Lowe, Nissan Pow, Iulian Serban, and Joelle Pineau. The ubuntu dialogue corpus: A large dataset for research in unstructured multi-turn dialogue systems. *arXiv preprint arXiv:1506.08909*, 2015.
- [55] Zhongqi Lu and Qiang Yang. Partially observable markov decision process for recommender systems. *arXiv preprint arXiv:1608.07793*, 2016.
- [56] Tariq Mahmood and Francesco Ricci. Improving recommender systems with adaptive conversational strategies. In *Proceedings of the 20th ACM conference on Hypertext and hypermedia*, pages 73–82. ACM, 2009.
- [57] François Mairesse, Milica Gašić, Filip Jurčiček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1552–1561. Association for Computational Linguistics, 2010.
- [58] Christopher Manning, Prabhakar Raghavan, and Hinrich Schütze. Introduction to information retrieval. *Natural Language Engineering*, 16(1):100–103, 2010.
- [59] Julian McAuley and Alex Yang. Addressing complex and subjective product-related queries with customer reviews. In *Proceedings of the 25th International Conference on World Wide Web*, pages 625–635. International World Wide Web Conferences Steering Committee, 2016.
- [60] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. On the dynamic generation of compound critiques in conversational recommender systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 176–184. Springer, 2004.
- [61] David McSherry and David W Aha. Avoiding long and fruitless dialogues in critiquing. In *International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 173–186. Springer, 2006.
- [62] Michael F McTear. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)*, 34(1):90–169, 2002.

- [63] Grégoire Mesnil, Yann Dauphin, Kaisheng Yao, Yoshua Bengio, Li Deng, Dilek Hakkani-Tur, Xiaodong He, Larry Heck, Gokhan Tur, Dong Yu, et al. Using recurrent neural networks for slot filling in spoken language understanding. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):530–539, 2014.
- [64] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.
- [65] Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.
- [66] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.
- [67] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.
- [68] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.
- [69] Raymond J Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the fifth ACM conference on Digital libraries*, pages 195–204. ACM, 2000.
- [70] Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*, 2015.
- [71] Nikola Mrkšić, Diarmuid O Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. Neural belief tracker: Data-driven dialogue state tracking. *arXiv preprint arXiv:1606.03777*, 2016.
- [72] Karthik Narasimhan, Adam Yala, and Regina Barzilay. Improving information extraction by acquiring external evidence with reinforcement learning. *arXiv preprint arXiv:1603.07954*, 2016.

- [73] Alice H Oh and Alexander I Rudnicky. Stochastic language generation for spoken dialogue systems. In *ANLP-NAACL 2000 Workshop: Conversational Systems*, 2000.
- [74] Michael Pazzani and Daniel Billsus. Learning and revising user profiles: The identification of interesting web sites. *Machine learning*, 27(3):313–331, 1997.
- [75] Michael J Pazzani and Daniel Billsus. Content-based recommendation systems. In *The adaptive web*, pages 325–341. Springer, 2007.
- [76] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, Kam-Fai Wong, and Shang-Yu Su. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. *arXiv preprint arXiv:1801.06176*, 2018.
- [77] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016.
- [78] Antoine Raux, Brian Langner, Dan Bohus, Alan W Black, and Maxine Eskenazi. Let’s go public! taking a spoken dialog system to the real world. In *Ninth European conference on speech communication and technology*, 2005.
- [79] Siva Reddy, Danqi Chen, and Christopher D Manning. Coqa: A conversational question answering challenge. *arXiv preprint arXiv:1808.07042*, 2018.
- [80] James Reilly, Barry Smyth, Lorraine McGinty, and Kevin McCarthy. Critiquing with confidence. In *International Conference on Case-Based Reasoning*, pages 436–450. Springer, 2005.
- [81] Steffen Rendle. Factorization machines. In *2010 IEEE International Conference on Data Mining*, pages 995–1000. IEEE, 2010.
- [82] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.
- [83] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186. ACM, 1994.
- [84] Francesco Ricci and Tariq Mahmood. Learning and adaptivity in interactive recommender systems. In *ECAI 2006 Workshop on Recommender Systems*, page 111, 2006.

- [85] Francesco Ricci and Quang Nhat Nguyen. Critique-based mobile recommender systems. *OEGAI Journal*, 24(4):1–7, 2005.
- [86] Francesco Ricci, Lior Rokach, and Bracha Shapira. Introduction to recommender systems handbook. In *Recommender systems handbook*, pages 1–35. Springer, 2011.
- [87] Verena Rieser and Oliver Lemon. *Reinforcement learning for adaptive dialogue systems: a data-driven methodology for dialogue management and natural language generation*. Springer Science & Business Media, 2011.
- [88] Alan Ritter, Colin Cherry, and Bill Dolan. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics, 2010.
- [89] Pornthep Rojanavas, Phaitoon Srinil, and Ouen Pinngern. New recommendation system using reinforcement learning. *Special Issue of the Intl. J. Computer, the Internet and Management*, 13(SP 3), 2005.
- [90] Nicholas Roy, Joelle Pineau, and Sebastian Thrun. Spoken dialogue management using probabilistic reasoning. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 93–100. Association for Computational Linguistics, 2000.
- [91] Alexander Rudnicky and Wei Xu. An agenda-based dialog management architecture for spoken language systems. In *IEEE Automatic Speech Recognition and Understanding Workshop*, volume 13, 1999.
- [92] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2015.
- [93] Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *arXiv preprint arXiv:1603.06807*, 2016.
- [94] Iulian Vlad Serban, Ryan Lowe, Peter Henderson, Laurent Charlin, and Joelle Pineau. A survey of available corpora for building data-driven dialogue systems: The journal version. *Dialogue & Discourse*, 9(1):1–49, 2018.
- [95] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871*, 2018.

- [96] Guy Shani, David Heckerman, and Ronen I Brafman. An mdp-based recommender system. *Journal of Machine Learning Research*, 6(Sep):1265–1295, 2005.
- [97] Upendra Shardanand and Pattie Maes. Social information filtering: algorithms for automating "word of mouth". In *Chi*, volume 95, pages 210–217. Citeseer, 1995.
- [98] Sybil Shearin and Henry Lieberman. Intelligent profiling by example. In *Proceedings of the 6th international conference on Intelligent user interfaces*, pages 145–151. ACM, 2001.
- [99] Yue Shi, Martha Larson, and Alan Hanjalic. List-wise learning to rank with matrix factorization for collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 269–272. ACM, 2010.
- [100] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.
- [101] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *ICML*, 2014.
- [102] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.
- [103] Amanda Stent, Rashmi Prasad, and Marilyn Walker. Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd annual meeting on association for computational linguistics*, page 79. Association for Computational Linguistics, 2004.
- [104] Xiaoyuan Su and Taghi M Khoshgoftaar. A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009, 2009.
- [105] Yueming Sun and Yi Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 235–244. ACM, 2018.
- [106] Yueming Sun, Yi Zhang, Yunfei Chen, and Roger Jin. Conversational recommendation system with unsupervised learning. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 397–398. ACM, 2016.

- [107] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.
- [108] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.
- [109] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.
- [110] Nima Taghipour, Ahmad Kardan, and Saeed Shiry Ghidary. Usage-based web recommendations: a reinforcement learning approach. In *Proceedings of the 2007 ACM conference on Recommender systems*, pages 113–120. ACM, 2007.
- [111] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In *Proceedings of the 1993 Connectionist Models Summer School Hillsdale, NJ. Lawrence Erlbaum*, 1993.
- [112] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.
- [113] Damir Vandic, Steven Aanen, Flavius Frasincar, and Uzay Kaymak. Dynamic facet ordering for faceted product search engines. *IEEE Transactions on Knowledge and Data Engineering*, 29(5):1004–1016, 2017.
- [114] Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.
- [115] Marilyn A Walker, Owen C Rambow, and Monica Rogati. Training a sentence planner for spoken dialogue using boosting. *Computer Speech & Language*, 16(3-4):409–433, 2002.
- [116] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.
- [117] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*, 2016.
- [118] Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*, 2015.

- [119] Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*, 2016.
- [120] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proceedings of the SIGDIAL 2013 Conference*, pages 404–413, 2013.
- [121] Jason D Williams and Steve Young. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422, 2007.
- [122] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.
- [123] Ji Wu, Miao Li, and Chin-Hui Lee. A probabilistic framework for representing dialog systems and entropy-based dialog management through dynamic stochastic state evolution. *IEEE/ACM Transactions on Audio, Speech and Language Processing (TASLP)*, 23(11):2026–2035, 2015.
- [124] Zhao Yan, Nan Duan, Peng Chen, Ming Zhou, Jianshe Zhou, and Zhoujun Li. Building task-oriented dialogue systems for online shopping. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.
- [125] Xuesong Yang, Yun-Nung Chen, Dilek Hakkani-Tür, Paul Crook, Xiujun Li, Jianfeng Gao, and Li Deng. End-to-end joint learning of natural language understanding and dialogue manager. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5690–5694. IEEE, 2017.
- [126] Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179, 2013.
- [127] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. Personalized entity recommendation: A heterogeneous information network approach. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 283–292. ACM, 2014.
- [128] Jiyong Zhang, Nicolas Jones, and Pearl Pu. A visual interface for critiquing-based recommender systems. In *Proceedings of the 9th ACM conference on Electronic commerce*, pages 230–239. ACM, 2008.

- [129] Jiyong Zhang and Pearl Pu. A comparative study of compound critique generation in conversational recommender systems. In *International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems*, pages 234–243. Springer, 2006.
- [130] Lanbo Zhang and Yi Zhang. Interactive retrieval based on faceted feedback. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 363–370. ACM, 2010.
- [131] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too? *arXiv preprint arXiv:1801.07243*, 2018.
- [132] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):5, 2019.
- [133] 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 27th ACM International Conference on Information and Knowledge Management*, pages 177–186. ACM, 2018.
- [134] Tiancheng Zhao and Maxine Eskenazi. Towards end-to-end learning for dialog state tracking and management using deep reinforcement learning. *arXiv preprint arXiv:1606.02560*, 2016.
- [135] Xiangyu Zhao, Long Xia, Liang Zhang, Zhuoye Ding, Dawei Yin, and Jiliang Tang. Deep reinforcement learning for page-wise recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 95–103. ACM, 2018.
- [136] Guanjie Zheng, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. Drn: A deep reinforcement learning framework for news recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 167–176. International World Wide Web Conferences Steering Committee, 2018.