

Santa Clara University

Scholar Commons

Engineering Ph.D. Theses

Student Scholarship

5-2022

Personalized Memory Transfer for Conversational Recommendation Systems

Naga Archana Godavarthy

Follow this and additional works at: https://scholarcommons.scu.edu/eng_phd_theses



Part of the [Computer Engineering Commons](#)

SANTA CLARA UNIVERSITY

Department of Computer Science and Engineering

Date: May 2022

I HEREBY RECOMMEND THAT THE THESIS PREPARED UNDER

DR. YI FANG

BY

NagaArchana Godavarthy

ENTITLED

Personalized Memory Transfer for Conversational
Recommendation Systems

BE ACCEPTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE
OF

DOCTOR OF PHILOSOPHY IN COMPUTER SCIENCE AND
ENGINEERING



Yi Fang (May 24, 2022 14:17 PDT)

Thesis Advisor
Dr. Yi Fang



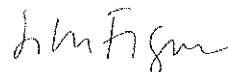
Ahmed Amer (May 27, 2022 09:10 PDT)

Thesis Reader
Dr. Ahmed Amer

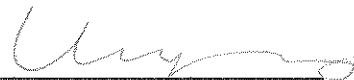


N. Ling (May 27, 2022 09:26 PDT)

Chairman of Department
Dr. Nam Ling



Thesis Reader
Dr. Silvia Figuerira



Thesis Reader
Dr. Weijia Shang



Thesis Reader
Dr. Nicholas Tran

Personalized Memory Transfer for Conversational Recommendation Systems

by

NagaArchana Godavarthy

Dissertation

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Computer Science and Engineering
in the School of Engineering at Santa Clara University, 2022

Santa Clara, California



To my Late Mother Smt.K.Sai Leela.

Acknowledgements

I would like to acknowledge several people who supported the creation of this work.

First and foremost, I would like to express my sincere gratitude for my advisor Dr.Yi Fang for accepting me as his Ph.D. student. He always believed in me and gently pushed me to do better at every step. He has supported me through many tough times in my Ph.D. journey. He has been a great mentor and role model not only in academics, but also in life in general. He gracefully demonstrated many wonderful qualities like perseverance, curiosity, kindness, hard-work. I hope to integrate and incorporate those qualities myself.

It was my husband, Syam who first broached this idea of working towards Ph.D. He believed in me and supported me throughout the journey and believed it would be a great example for our daughters, Lasya and Lahari. Many thanks and much love to my daughters who patiently allowed me to continue working on my Ph.D. I am also immensely grateful to my parents, Sai Leela, Sri Hari and my in-laws Visalakshi, Surya Bhagavanulu for supporting us in this endeavour. Thanks to my siblings Aparna and Abishek for their support.

I would like to thank my lab mates, Travis, Yuan, Suthee, Xuyang to name a few who helped and supported me at various stages of the process.

I also thank Late Dr.Joanne Holiday, Dr.Ahmed Amer, Dr.Nicolas Tran and Dr.Christopher Kitts for giving me time and letting me explore and expand my interests.

I would like to thank my doctoral committee Dr.Silvia Figueira, Dr.Ahmed Amer, Dr.Nicholas Tran, Dr.Weija Shang for their time. I would also like to thank the Dean for giving multiple chances for my journey.

Finally I am grateful to all the people who wished me best. I have to acknowledge the life lessons I have learned during the entire PhD journey along with life experiences. With an open mindset, all experiences in life can be turned into opportunities to learn something useful, which makes life a life-long journey. Multiple factors play a role towards success of any great accomplishment. Although intelligence plays a major role in academic success, that by itself is not sufficient. Hard-work and humility are also equally important if not more. "Never Give Up", although sounds like a cliched quote, it has deep meaning. Many life situations force people to give-up. But if "Never Give Up" attitude is adopted even at the level of intention, I believe, sooner or later nature will make things happen.

My Mother has been a great example and Role model throughout her life showing me "Never Give Up" attitude by example till the very end. I believe and 🙏 that the efforts are never wasted.

I sincerely hope I can pay forward all the lessons I have learned and have an open-mind to continue learning from my successors.

Sincerely,

Personalized Memory Transfer for Conversational Recommendation Systems

NagaArchana Godavarthy

Department of Computer Science and Engineering

Santa Clara University

Santa Clara, California

2022

ABSTRACT

Dialogue systems are becoming an increasingly common part of many users' daily routines. Natural language serves as a convenient interface to express our preferences with the underlying systems. In this work, we implement a full-fledged Conversational Recommendation System, mainly focusing on learning user preferences through online conversations. Compared to the traditional collaborative filtering setting where feedback is provided quantitatively, conversational users may only indicate their preferences at a high level with inexact item mentions in the form of natural language chit-chat. This makes it harder for the system to correctly interpret user intent and in turn provide useful recommendations to the user. To tackle the ambiguities in natural language conversations, we propose Personalized Memory Transfer (PMT) which learns a personalized model in an online manner by leveraging a key-value memory structure to distill user feedback directly from conversations. This memory structure enables the integration of prior knowledge to transfer existing item representations/preferences and natural language representations. We also implement a retrieval based response generation module, where the system in addition to recommending items to the user, also

responds to the user, either to elicit more information regarding the user intent or just for a casual chit-chat. The experiments were conducted on two public datasets and the results demonstrate the effectiveness of the proposed approach.

Contents

Acknowledgements	iv
Abstract	vi
Contents	viii
List of Figures	x
List of Tables	xii
1 Introduction	11
1.1 Introduction	11
2 Related Work	16
2.1 Conversational Recommendation Systems	16
2.2 Task-Oriented Dialog System	19
2.3 Cross-domain Recommendation	20
3 Foundation	23
3.1 Conversational Recommendation Systems	23
3.1.1 Motivation	24
3.1.2 History	24
3.1.3 Underlying Knowledge and Data	26
3.1.4 Computational Tasks	28
3.1.4.1 Main Tasks	28
3.1.4.2 Supporting Tasks	30
3.2 Memory Networks	30
3.3 Latent Factor Models	35
3.3.1 Collaborative Filtering	35
3.3.1.1 Memory-Based Collaborative Filtering	37

3.3.1.2	Model-Based Colaborative Filtering	39
	Non-Negative Matrix Factorization	40
	Generalized Matrix Factorization	41
4	Methodology	42
4.1	Key-Value Memory	44
4.2	Key Addressing	46
4.3	Key-Value Reading	47
4.4	Interaction Function	49
4.5	Learning the User Preference Vector	51
4.6	Recommendation	53
4.7	Response Generation	53
5	Experiments	55
5.1	Datasets	55
5.2	Evaluation Metrics	56
5.3	Baselines and Settings	57
5.4	Baseline Comparison	61
5.5	Effect of Sentence Encoder	64
5.6	Effect of Embedding Size	66
5.7	Effect of Conversation Length	67
5.8	Effect of Number of Layers for Different Embedding Sizes	69
5.9	Efficiency Analysis	70
5.10	Qualitative Study	72
5.11	Response Generation	75
6	Conclusion and Future Work	80
6.1	Conclusion	80
6.2	Future Work	81
6.2.1	Joint Optimization of Different Modules	81
6.2.2	Explainability	81
6.2.3	Sophisticated Strategies for Multi-turn Conversation	82
6.2.4	Knowledge Enrichment	83
	Bibliography	84

List of Figures

1.1	Example dialogue between the Seeker asking for recommendations and the Recommender providing suggestions. Movie mentions are in bold.	13
3.1	Typical architecture of a Conversational Recommender System	25
3.2	Main Computational Tasks of a Conversational Recommendation System	28
3.3	Interaction between Memory and Controller Modules	33
3.4	Question Answering using Memory Module	33
3.5	Factoid QA with Single Supporting fact	34
3.6	Factoid QA with Two Supporting fact	34
3.7	Types of Recommender Systems	36
3.8	Matrix Factorization	36
4.1	Illustration of PMT encoding the user’s utterance through the key-value memory structure producing the final output memory to estimate the interaction function for a new user.	43
5.1	Effect of embedding size on the proposed PMT-GMF model on the two datasets in different metrics	67
5.2	Recall at 25 (R@25) of different methods over the number of turns in the conversations.	68
5.3	Effect of number of layers with embedding size 10 on the proposed PMT-GMF model on the two datasets for Recall	70
5.4	Effect of number of layers with embedding size 20 on the proposed PMT-GMF model on the two datasets for Recall	70
5.5	Effect of number of layers with embedding size 30 on the proposed PMT-GMF model on the two datasets for Recall	71
5.6	Effect of number of layers with embedding size 40 on the proposed PMT-GMF model on the two datasets for Recall	71
5.7	an example conversation from the GoRecDial dataset between recommendation seeker (Seeker) and recommender (Rec). Marked in blue in the figure are the top 10 movie recommendations by PMT-GMF. The positive recommendation is marked in bold based on the ground truth.	73
5.8	Examples of Response Generation for a given Seeker dialog for Redial dataset.	76

5.9	Examples of Response Generation for a given Seeker dialog for Redial dataset.	77
5.10	Examples of Response Generation for a given Seeker dialog for GoRecdial dataset.	78
5.11	Examples of Response Generation for a given Seeker dialog for GoRecdial dataset.	79

List of Tables

5.1	Experimental results on the ReDial dataset for different methods reporting Precision (P), Recall (R), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) for different cut offs. The best results are highlighted in bold. † denotes the improvement over the best result of the baselines is statistically significant based on the paired t-test (p-value < 0.05).	61
5.2	Experimental results on the GoRecDial dataset for different methods reporting Precision (P), Recall (R), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) for different cut offs. The best results are highlighted in bold. † denotes the improvement over the best result of the baselines is statistically significant based on the paired t-test (p-value < 0.05).	63
5.3	Experimental results of PMT-GMF on the ReDial dataset with different sentence encoders. The best results are highlighted in bold.	65
5.4	Experimental results of PMT-GMF on the GoRecDial dataset with different sentence encoders. The best results are highlighted in bold.	66
5.5	Average response time (in seconds) of PMT-GMT with two different sentence encoders on the two datasets: BERT and USE-Transformer.	72

Chapter 1

Introduction

1.1 Introduction

Virtual assistants such as Amazon Alexa, Apple Siri, Google Assistant, and Microsoft Cortana are becoming an increasingly common part of many users' daily routines. Conversational agents are being quickly adopted in industry to handle customer service requests at banks, set up travel accommodations, and make product recommendations to online retailers. Natural language serves as a convenient interface to computing systems and is a natural form to express our preferences to others. Consequently, conversational recommendation systems have emerged to elicit the dynamic preferences of users through multi-turn interactions in natural language.

In the setting of conversational recommendations, two parties are interacting with one another centered around discussing items, e.g., movies [47]. The first party (seeker)

expresses his/her preferences and asks for relevant movie suggestions from the second party who acts as the recommender. This recommender’s goal is to understand the seeker and provide personalized movie recommendations based on the conversation. An example dialogue is shown in Figure 1.1. The conversational recommendation setting poses ambiguity where the user expresses feedback in the form of natural language, in contrast to the traditional collaborative filtering setting where feedback is typically provided in explicit (ratings) or implicit (clicks/views) form. The seeker may not specify which item(s) he or she likes but rather describes it at a high level such as “I would like to watch a suspenseful, but clean family friendly movie” in Figure 1.1. Even when the user makes specific references, items or movies mentioned in the dialogue may not be exactly accurate. For example, *A Space Odyssey* and *Star Wars: The Last Jedi* are not the full names of the movies (which should be *2001: A Space Odyssey (1968)* and *Star Wars: Episode VIII - The Last Jedi (2017)* respectively based on the IMDB database). Also, the *Star Wars* that the seeker mentioned is a movie franchise and cannot be mapped to a specific movie. Some prior work on conversational recommendations used synthetic data [22, 84, 44] or assumed an entity tagger mapping movie mentions in the dialogue to unique identifiers [47] which may not be available in practice.

An ideal conversational recommendation system would learn to adapt to user’s preferences as the conversation progresses. Inspired by key-value memory networks [61] originally proposed for question answering and the successful application of memory networks to the recommender domain [24, 34], we propose Personalized Memory Transfer (PMT) to learn user preferences in a natural chit-chat conversational setting. To

Seeker: Hi
Rec: Hello
Seeker: How are you? I would like to watch a suspenseful,
but clean family friendly movie.
Rec: Oh! I love suspenseful movies. Have you seen **The Illusionist**, it is
pg-13 however
Seeker: We have not seen that. We enjoyed **A Space Odyssey**
Rec: If you like **A Space Odyssey**, you can also try **Star Wars: The Last
Jedi** for something G Rated
Seeker: Honestly I'm not a big fan of **Star Wars**. I mean they are great
movies but for some reason i just don't like it
Rec: You can try **Planet of the Apes** the older one is quite suspenseful
and family friendly.
Seeker: Those sound good! I'm going to look into those movies.
Rec: I hope you enjoy, have a nice one.
Seeker: Thank you for your help! Have a great night! Good bye

FIGURE 1.1: Example dialogue between the Seeker asking for recommendations and the Recommender providing suggestions. Movie mentions are in bold.

tackle the ambiguity in natural language, we treat each conversation as a virtual item by combining the known item representations. As the conversation progresses, PMT updates the user representation based on the observed virtual items. Since collecting large amounts of conversational data with labels may be cost prohibitive or infeasible due to privacy concerns, we propose to leverage two forms of transfer learning to address data sparsity.

First, we learn an interaction function measuring the user's level of interest in a given item from a large-scale collaborative filtering dataset (e.g. MovieLens [31]) and transfer the learned movie representations to the conversational recommendation domain. It is worth noting that the set of users in the interaction function are specific to the MovieLens dataset, different from the users for the Conversational Recommendation task. Next, each movie is represented by a memory slot with a corresponding pair of key and value

memory in the key-value memory structure. This memory structure is tailored such that the model uses the keys to address relevant memories with respect to the current user’s utterance (query) followed by a reading phase which returns the final output memory using the value memory which serves as a virtual item. A pre-trained natural language encoder maps each movie’s plot to a low-dimensional representation and the learned item preferences act as the basis for the key and value memories, respectively. During the conversation the user’s utterance is mapped to a low-dimensional semantic vector with the natural language encoder which addresses the key memories by identifying relevant movies for the user. For each conversation we learn a new user representation in an online manner by leveraging the virtual item as a positive instance along with considering user’s satisfaction to evaluate the pairwise objective. As the conversation progresses PMT updates the user representation with standard backpropagation via stochastic gradient descent guided by the virtual item extracted and sentiment from the user’s utterance permitting personalized movie recommendations.

It is worth noting that a full-fledged conversational recommendation system should include a response generation component, which is to generate human-understandable responses for communicating with users and making recommendations. There exist multiple strategies and active research on how to generate readable, fluent, and consistent natural language responses [28]. In this work, in addition to learning user preferences over conversations, we also implement response generation using retrieval based model[58]. Our contributions can be summarized as follows:

- To the best of our knowledge, this is the first work that learns user preferences in an online fashion from conversations where there may only exist high level user feedback or inexact item mentions.
- We propose a novel memory network named Personalized Memory Transfer (PMT) to distill user feedback directly from user’s utterances by integrating prior knowledge of existing item representations/preferences and pre-trained language models, which allows the learning of a new user representation over conversations.
- We implement two simple retrieval-based response generation modules. This makes our implementation, a complete conversational recommendation system.
- Experimental results on two public conversational datasets demonstrate the effectiveness of PMT to learn a personalized user representations with two interaction functions and natural language encoders in an online setting. We made the source code publicly available on GitHub¹.

The remainder of the paper is organized as follows: Section 2 introduces the research status and related work. Section 3 lays the foundation for the main concepts used in our implementation. Section 4 presents the proposed approach. Section 5 demonstrates and analyzes our experimental results. Finally, Section 6 concludes the paper with a discussion on future work.

¹<https://github.com/agodavarthy/PMT>

Chapter 2

Related Work

2.1 Conversational Recommendation Systems

Recommendation systems are vital to keeping users engaged and satisfied with personalized recommendations in the age of information explosion. Modern E-commerce, entertainment and social media platforms provide personalized content by analyzing user preferences based on explicit and/or implicit feedback and infer their potentially preferred items. Based on the type of input data, the recommender systems can be summarized into collaborative filtering systems, content-based recommender systems, and hybrid recommender systems [102]. Early researches in collaborative filtering formulated the recommendation task as predicting the user rating score on the candidate items [82]. The rating-based recommendation models did not perform well in top- n recommendation, which motivated the ranking-based recommendation by learning a model

based on the relative preference of a user over pairs of items [76]. Content-based recommendation utilizes item or user features (such as item description, user profiles, and attributes) to find similar items or users for recommendations [54].

Most conventional recommender systems highly rely on users' historical trajectories to generate personalized recommendation. It lacks the capability of capturing users' dynamic intentions/demands. This intrinsic limitation motivates online recommendation with its goal to adapt the recommendation results with the user's online actions [48]. Much existing work models it as a multi-arm bandit problem [92, 97]. While achieving remarkable progress, the bandit-based solutions are still insufficient especially in the warm start scenarios [44]. In the recent years, conversational recommender systems have attracted an increased attention in the research community as they enable a system to interact with users using natural language. The general idea of such systems is to support a task-oriented, multi-turn dialogue with their users.

Sun and Zhang [85] represented a dialogue vector as a set of facet-value pairs and fuses the vector with user and item ratings via a factorization machine. Li et al. [47] assumed that movies mentioned in the conversation are known in advance. The model encodes the incoming utterances and progressively constructs the input vector to the user-based autoencoder to predict ratings. Our work differs from the existing works as our model does not need any prior knowledge of movies. Unlike [47], our work does not need to tag a movie directly in the conversation. A similar but related approach is interactive recommendation which elicits specific questions to the user regarding their preferences but may diminish user experience [14, 15, 107]. [64] estimates the value for each question

in the conversation to make sure that questions asked by the agent are relevant to the target item using actor-critic framework. [4] proposes a conversational paradigm for product search driven by non-relevant items, based on which fine-grained feedback is collected and utilized to show better results in the next iteration. [45] propose an interactive path reasoning algorithm on a heterogeneous graph on which users, items, and attributes are represented as nodes and an edge connected two nodes represented a relationship between two nodes, e.g., a user purchased an item, or an item has a certain value for an attribute. With the help of the graph, a conversation can be converted to a path on the graph.

There are many scenarios in the CRS, where the recommender has to respond appropriately to the user utterance. Some user utterance require preference refinement e.g., *I like Pulp Fiction, but not Quentin Tarantino* [67], while others require systems opinion about a particular item, *How about Huawei P9* [98]. In [72] the system responses are pooled from a Q&A knowledge base, which then are ranked to determine an appropriate response. In case of absence of suitable response, it is then generated by the sequence-to-sequence model. Another related work for response generation [69] trained two different RNNs, one to respond with general greetings or chit-chat and the other to respond to more specific user questions. When the system is unable to comprehend user intent, certain approaches like politeness, apology strategy, repetition/clarification are adopted [1]. For a comprehensive literature review on conversational recommendation systems, readers can refer to two recent survey articles [36, 28].

2.2 Task-Oriented Dialog System

A task-oriented dialog system’s intent is to assist users with a given task through natural language such as hotel booking, travel or online shopping. Task-oriented dialogue systems is one important branch in dialogue system research. Though conversational recommendation is an emerging research topic, many of the basic concepts and model designs were originated from task-oriented dialogue systems. Pioneering work by [98] leveraged natural language processing and crowdsourcing to build a dialog system for E-commerce. Bordes et al. [5] used a memory network to store user utterances and predicts the response for a restaurant reservation task. Wen et al. [95] modeled a task-oriented conversation as a sequence-to-sequence mapping problem while augmenting the model with the dialogue history. Mo et al. [63] learned a personalized dialogue agent by transferring relevant knowledge from conversations via reinforcement learning framework. For non-textual utterance such as images, Cui et al. [17] proposed a multi-modal encoder which is a combination of image and text encoders to jointly transform an utterance into a dialog vector. Zhang et al. [104] proposed the “systems ask, users respond” paradigm for conversational recommendation in e-commerce. A system agent is designed to ask users different questions to obtain clarified demands continuously, and a multi-memory network is utilized to analyze the user utterances for recommendation.

Understanding user preferences and intentions from dialogues is the key requirement for conversational recommendation or dialog systems in general, since subsequent tasks such as response generation heavily rely on this information. Much existing work focused on

the multi-turn strategy and core recommendation logic [28], while they circumvented the extraction of user preferences from raw natural language utterances and often required the preprocessed input such as rating scores [109] and YES/NO answers [108], which is unnatural in real-world human conversations. Some recent work extracted semantic information in users' raw utterance by utilizing deep learning, e.g., Li et al. [47] based on recurrent neural network (RNN), Liu et al. [51] based on convolutional neural network (CNN), and Penha and Hauff [70] based on the bidirectional encoder representations from transformers (BERT) [20].

[55] models two separate systems within a unified framework, seek high-level mapping between hierarchical dialog acts and multi-hop knowledge graph reasoning. The model walks on a large-scale knowledge graph to form a reasoning tree at each turn, then mapped to dialog acts to guide response generation. [74] propose the Knowledge-Based Question Generation System (KBQG), a framework for conversational recommendation which models a user's preference in a finer granularity by identifying the most relevant relations from a structured knowledge graph (KG).

2.3 Cross-domain Recommendation

Cross-domain recommendation seeks to transfer knowledge from a data-rich source domain and utilize it in a new target domain [78]. This may help alleviate the cold-start

problem or lack of sufficient data for personalized recommendations in the new target domain. Typical cross-domain recommendation models are extended from single-domain recommendation models [46]. Hu et al. [34] introduced a memory component jointly with a transfer network to selectively transfer source content information to the target domain for a given user. These approaches assume that different patterns characterize the way that users interact with items of a certain domain. However, our approach differs that it does not require users to be overlapped in the two domains and can address the user cold-start problem. Gao et al. [27] proposed two levels of attention mechanisms to transform the interaction history of a user in the source domain as an additional user latent factor in the target domain. Their approach exploits the transferable information from the set of overlapped items which may not be feasible to apply to the conversational domain where the source and target domains are often multimodal. Another recent approach by [37] uses domain separation networks [6] to train a multi-class classifier from the source domain to predict the preferred item in the target domain, but the idea may not be applicable to natural language conversations. [103] proposes Cycle Generation Networks (CGN) focuses on learning explicit mapping between a user's behaviors (i.e. interaction itemsets) in different domains during the same temporal period. [106] constructs two separate heterogeneous graphs based on the rating and content information from two domains to generate more representative user and item embeddings. Then, we propose an element-wise attention mechanism to effectively combine the embeddings of common users learned from both domains. [50]

proposes a novel Bi-directional Transfer learning method for cross-domain recommendation by using Graph Collaborative Filtering network as the base model (BiTGCF). [93] proposes a TagCDCTR (Tag-informed Cross Domain Collaborative Topic Regression) model, which exploits shared tags as bridges to link related domains through an extended collaborative topic modeling framework. The model exploits the inter-domain relations by encoding cross domain item-item similarity based on common tags and jointly learning a shared set of topics from all domains together.

Chapter 3

Foundation

3.1 Conversational Recommendation Systems

Recommender Systems utilize users historical preferences in the form of either explicit or implicit feedback to provide recommendations. However this kind of static recommendation has one major shortcoming, that the user preferences might be stale. Conversational recommendation systems can mitigate this problem to a certain extent, as they can obtain most recent user preferences using a dialog based conversation. Conversational Recommendation Systems provide personalized recommendation and natural language dialogue for the users, using either text, speech or vision based interface.

In this section, we will provide overview of Conversational Recommendation Systems.

3.1.1 Motivation

Typically, a recommender system presents a tailored set of recommendations after observing the users behavior over a period of time. Although such an approach is common and useful in various domains, it can have a number of potential limitations. There are, for example, a number of application scenarios, where the user preferences cannot be reliably estimated from their past interactions. This is often the case where the user even might have no past experience. The set of recommendations can be highly context-dependent, and it might be difficult to automatically determine the user's current situation or needs. Another assumption is that users already know their preferences when they arrive at the site which is not necessarily true. Users might also construct their preferences only during the decision process, when they become aware of the space of the options. In some cases, they might also learn about the domain and the available options only during the interaction with the recommender. Conversational Recommendation System addresses many of the above mentioned challenges.

3.1.2 History

Although Conversational Recommendation Systems is in boom right now, the main concept has started about 30 years ago, in IR(Information Retrieval), HCI(Human Computer Interaction), RecSys(Recommendation Systems) communities.

In [16], the authors proposed I^3R , one of the earliest information systems that enables user-system interaction through dialogue. Later [3], proposed MERIT, an interactive

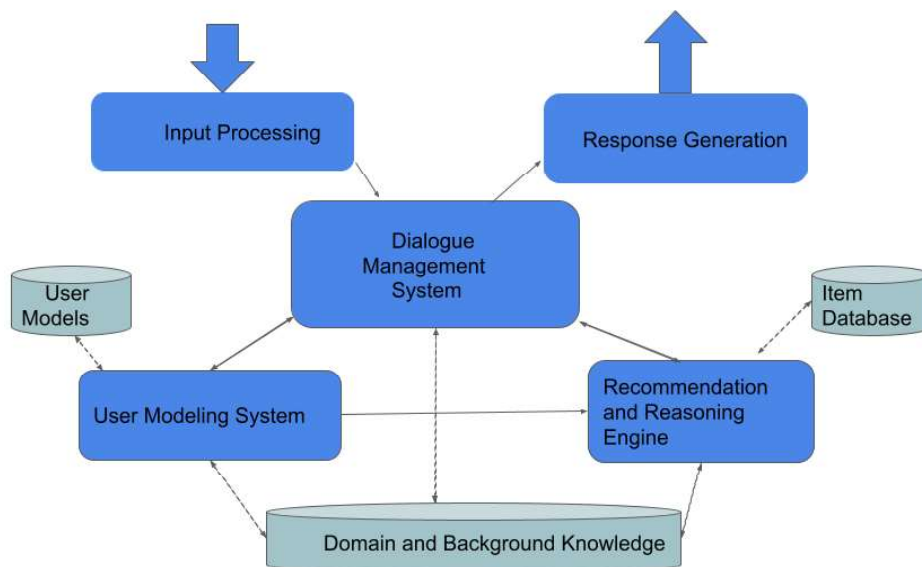


FIGURE 3.1: Typical architecture of a Conversational Recommender System

information seeking system by using script-based conversational interaction. In 2000 [30] formally introduced Conversational Recommendation System, using it for place recommendation. For more than 15 years after that the research in this area has been steadily growing. In 2018, there has been a surge of research work in this field, owing to the increasing popularity of Deep Learning and Reinforcement Learning.

Conversational Recommendation Systems can be formulated as a recommendation systems which can elicit dynamic user preferences and utilize the gained knowledge to satisfy user needs in a real-time multi-turn dialogue. A typical CRS architecture [36] include modules like input processing, output generation, dialogue management, user modeling and recommendation explanation module as shown in Figure 3.1. The input is obtained in many different modes including form-based, or natural language e.g., text, speech, vision. The input processing module handles the processing of the input

based on type of the input. One of the main features of Conversational Systems is the interaction initiative, including system-driven, user-driven and mixed-initiative systems.

Critiquing is a method of conversational recommendation that iteratively adapts recommendations in response to user preference feedback. Most of the critiquing-based systems are considered to be system-driven or mixed-initiative. The exclusively system-driven applications are form-based systems, where the system guides the user through a personalized preference elicitation until enough is known about the user to make recommendations. The other extreme is user-driven, which is mode like a question-answer mode, where “user-asks, system responds”.

The User Modeling System module learns user preferences and builds user profile during the conversation. The Recommender and Reasoning System retrieves a set of recommendations based on the current dialog state and current user profile. The response generation module is responsible for generating the system dialogue as a response. Various knowledge entities like item database, user models and background knowledge could be utilized for improving the overall performance.

3.1.3 Underlying Knowledge and Data

CRS is a dialog-based system, which has to support user’s information needs and user intent. Especially in NLP-based approaches, detecting user intent is one of the main

tasks. Some of the common domain-independent user-intents found in literature include starting/re-starting/ending a dialog, chit-chat, obtaining recommendations and explanations.

User Modeling is one of the central tasks of CRS. User preferences can be elicited by various modalities and forms in which user might want to express the preferences. The modeling of user using the preferences can be utilized for further inferences regarding the relevance of the recommended items. User preferences can be obtained in two ways, during the ongoing dialog (short-term) or past user preferences or preference over multiple sessions (long-term). Some of the ways user preferences could be elicited from the on-going dialog is individual item ratings(likes, dislikes) or from tags, key-phrases, etc.

CRS systems utilize additional knowledge related to items to be recommended(movies, books, etc) like corpora of the natural language conversation for learning and additional knowledge sources for entity recognition. Item related knowledge include item-ratings, meta-data including attributes of the item(e.g., genre of movies). These attributes can be utilized to compute personalized recommendation or provide explanations for the recommendations. Examples of some item related knowledge are MovieLens, Netflix ratings, amazon electronic products reviews, etc. CRS dialogue corpora are usually built using recorded and annotated conversation between human annotators. Some of the dataset built that way include ReDial, GoRecDial, bAbI, etc. In our work we used the ReDial and GoRecDial datasets.

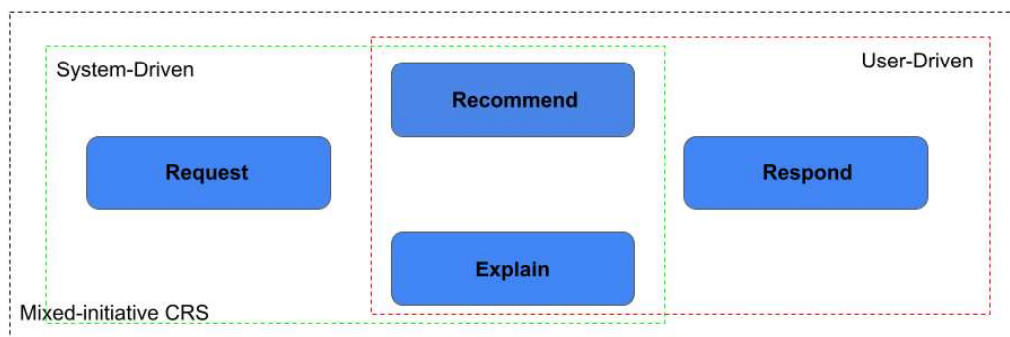


FIGURE 3.2: Main Computational Tasks of a Conversational Recommendation System

3.1.4 Computational Tasks

3.1.4.1 Main Tasks

CRS carry out four general types of tasks during conversations : Request, Recommend, Explain, and Respond, as seen in Figure 3.2.

Request: One main task of CRS is to determine next question to ask, preferably with increased dialog efficiency, i.e., with fewer number of turns. Many of the CRS models use “slot-filling” approach, seeking to acquire user preference information regarding predefined item attributes. Early CRS systems used heuristics to rank the item attributes for which the user has not expressed any preference yet [101, 87, 83]. Some of the approaches include user feedback on a pair of items or set of items [52], popularity, diversity of items [8, 60, 67, 73]. More recent work explore learning based approaches including reinforcement, RNN-based learning [85, 88, 56].

Recommend: This is the pivotal task of a CRS system. There are many approaches including the traditional collaborative filtering, content-based, knowledge-based, hybrid [53, 68, 94]. In [80], a history-guided critiquing system was proposed that aims to retrieve recommendation candidates from other users' sessions that are similar to the one of the current user. In [21], a travel recommender system was implemented that computes recommendations based on the relevance of item attributes to user preferences based on the Euclidean Distance.

Explain: Explanations can increase the system's perceived transparency, user trust and satisfaction, and they can help users make faster and better decisions [29].

Respond: This task is relevant to user-initiated or mixed NLP based CRS system where user actively asks a question, issues a command to the system. This task related to situations which are not recommendations or explanations. Two approaches can be used to generate system response for user utterances. One approach is to map the utterances to predefined user intents and then system responds using templates. Alternative approach is to automatically generate systems response using machine learning model from dialog corpora. Some of the conversation breakdown approaches are adopted in situations where system cannot infer user intent, by apologizing and politely asking to repeat the question.

In literature commonly used response generation approaches include retrieval-based and generation-based. [58] compares both the retrieval-based and generation-based and showed that KBR(Knowledge-based Retrieval model) can be favorable over the deep

learning based generation models from user perspective. They used crowd source to evaluate if the responses generated by both retrieval-based and generation-based methods are relevant to the user utterance in the the dialogue.

3.1.4.2 Supporting Tasks

In NLP-based CRS, it is critical to understand user intent, based on which system needs to take an appropriate action. Some of the supporting tasks include intent detection(including sentiment analysis) and named entity recognition(NER) [65].

In our work, we implement the many of the modules including input processing, dialog management system, user modeling, recommendation modules and response generation. However the main focus of this work is to learn the user profile in online fashion as the conversation unfolds. We will provide more details on the modules in the Chapters 4 and 5.

3.2 Memory Networks

Memory Networks are a class of models that combine large memory with learning components that can read and write to it [96]. Early Deep learning models like RNN, LSTM, GRU have capability for short term memory. Recurrent Neural Networks(RNN) [100] models are very effective for many applications including neural machine translation [2, 13], speech recognition [49], generation image descriptions [40], etc. Most of these

involve dealing with a single sentence. But when it comes to analyzing multiple sentences to arrive at the solution, more complex models including attention [90] or memory [43, 11] are required.

However when there is some type of dependencies within the data, i.e., out-of-order access or long-term dependency, LSTM, GRU models do not suffice. The main motivation for Memory Networks is long term memory. Long term memory is required for tasks like reading comprehension where in some questions are answered after reading a story. The model incorporates Reasoning with Attention over Memory(RAM).

The main components of memory network as implemented in [96] are:

- **I** (input): converts to bag-of-word-embeddings \mathbf{x} .
- **G** (generalization): stores \mathbf{x} in next available slot m_n .
- **O** (output): Loops over all memories $k=1$ or 2 times:
 - 1st loop max: finds best match m_i with \mathbf{x} .
 - 2nd loop max: finds best match m_j with (\mathbf{x}, m_i) .
 - The output \mathbf{o} is represented with (\mathbf{x}, m_i, m_j) .
- **R** (response): ranks all words in the dictionary given \mathbf{o} and returns best single word or full RNN.

Objective function: Minimize:

$$\begin{aligned} & \sum_{\bar{f} \neq m_{O1}} \max(0, \gamma - s_O(x, m_{O1}) + s_O(x, \bar{f})) + \\ & \sum_{\bar{f}' \neq m_{O2}} \max(0, \gamma - s_O([x, m_{O1}], m_{O2}) + s_O([x, m_{O1}], \bar{f}')) + \\ & \sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, m_{O1}, m_{O2}], r) + s_R([x, m_{O1}, m_{O2}], \bar{r})) \end{aligned}$$

Where:

S_O is the matching function for the Output component.

S_R is the matching function for the Response component.

x is the input question.

m_{O1} is the first true supporting memory (fact).

m_{O2} is the second true supporting memory (fact).

r is the response.

Figure 3.3 shows the interaction between the memory and controller module, which learn the new representations to the stored in the memory using the new input sentences.

[96] formulated a set of 20 tasks, similar to unit testing in computer science. Each “leaf” task is an independent task which can be tested one specific aspect of intended behavior. Subsequently “non-leaf” dependent tasks can be built by testing combinations.

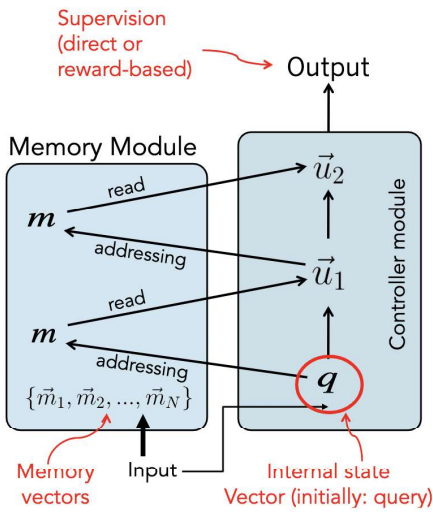


FIGURE 3.3: Interaction between Memory and Controller Modules

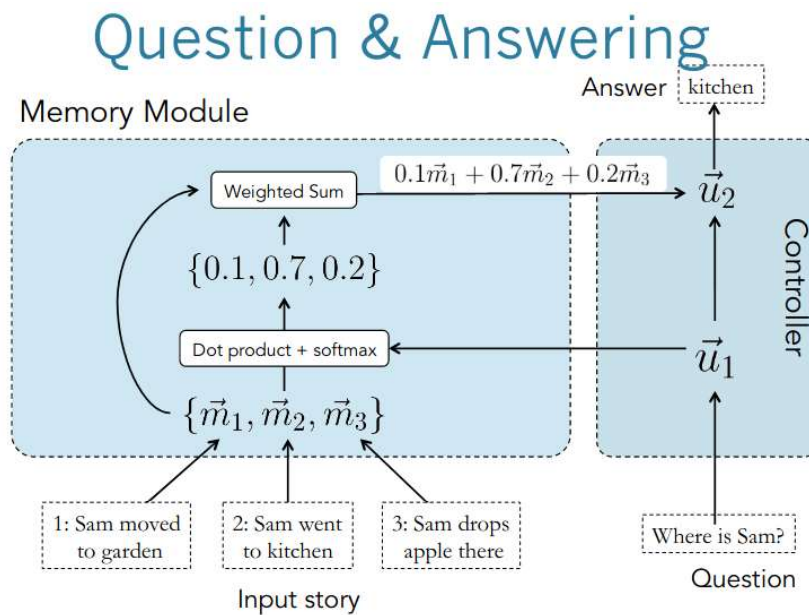


FIGURE 3.4: Question Answering using Memory Module

Figure 3.4 shows an example of internal working of memory module for a reading comprehension task where the system receives three input sentences and has to answer a question based on the input story. Figure 3.5 shows an example of a another factoid QA



FIGURE 3.5: Factoid QA with Single Supporting fact

with two supporting facts. In this case the question “Where is John” could be answered using one supporting fact of “John is in the playground.” which is two sentences farther.

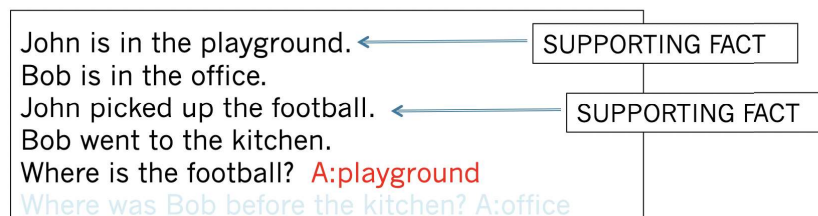


FIGURE 3.6: Factoid QA with Two Supporting fact

Figure 3.6 shows an example of a another factoid QA with two supporting facts. In this case the question “Where is the football?” could be answered using two supporting facts which are two(“John picked up the football”) and four(“John is in the playground”) sentences farther.

Similarly there are many more tasks including two argument relations: subject vs object, yes/no questions, counting, lists/sets, basic coreference, compound coreference, time manipulation, basic deduction, positional reasoning, reasoning about size, path finding, etc.

3.3 Latent Factor Models

3.3.1 Collaborative Filtering

A Recommendation System is a system which predicts the ratings the user might give a particular item. It is widely adopted in industry where companies recommend their products to the online customers e.g., Amazon(products), Google(Search, YouTube), etc. There are mainly 2 types of Recommender Systems, collaborative filtering [25], content-based filtering [89]. Collaborative filtering method is an application of Matrix Factorization [42] to find the relationship between items' and users' entities. Latent features, the association between users and items matrices, are determined to find similarity and make a prediction based on both item and user entities.

As shown in Figure 3.7, there are various types of Recommender Systems. On the high level there are collaborative filtering, content-based and hybrid of these two. There are three types of collaborative filtering including, memory-based, model-based and hybrid. Memory-based models include user-based similarity and item-based similarity. Model-based models include clustering(KNN), matrix factorization based(SVD, NMF) and deep learning based(GMF) As the Netflix Prize competition has demonstrated, matrix factorization models are superior to classic nearest-neighbor techniques for producing product recommendations, allowing the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels [42]. Matrix factorization models map both users and items to a joint latent factor space of dimensionality f , such

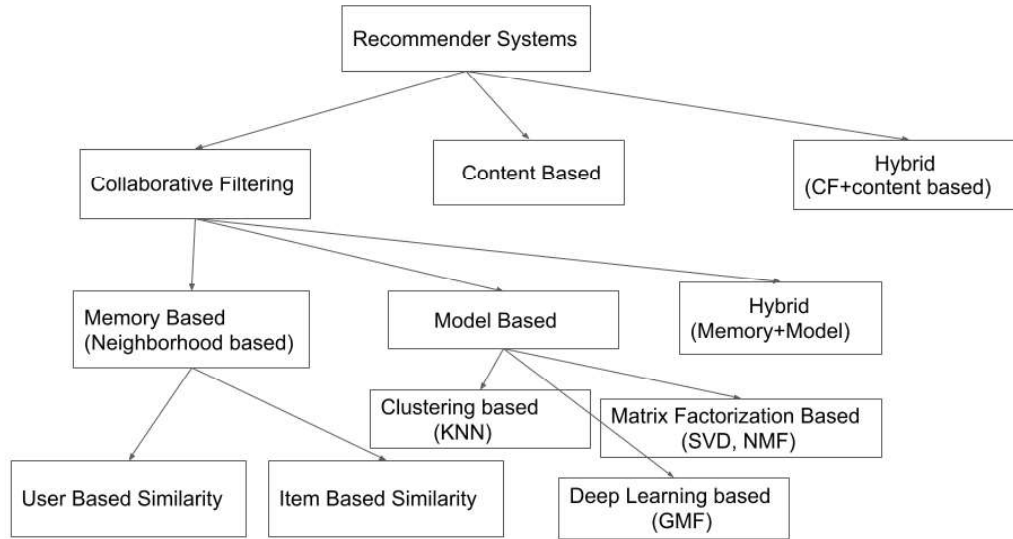


FIGURE 3.7: Types of Recommender Systems

that user-item interactions are modeled as inner products in that space. Given a set of \mathbf{U} users and \mathbf{I} items, the set of ratings from user for some items can be computed using Matrix Factorization as in Figure 3.8.

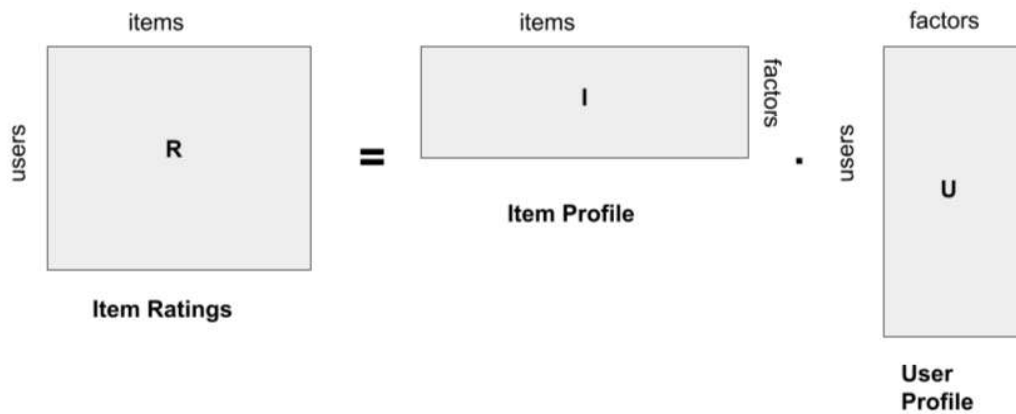


FIGURE 3.8: Matrix Factorization

3.3.1.1 Memory-Based Collaborative Filtering

The memory-based approach uses user rating data to compute the similarity between users or items. Typical examples of this approach are neighbourhood-based CF and item-based/user-based top-N recommendations.

For example, in user based approaches, the value of ratings user \mathbf{u} gives to item \mathbf{i} is calculated as an aggregation of some similar users' rating of the item: $r_{u,i} = \text{aggr}_{u' \in U} r_{u',i}$

where U denotes the set of top K users that are most similar to user \mathbf{u} who rated item \mathbf{i} . An example of the aggregation function can be:

$$r_{u,i} = \frac{1}{N} \sum_{u' \in U} r_{u',i}$$

The neighborhood-based algorithm calculates the similarity between two users or items, and produces a prediction for the user by taking the weighted average of all the ratings. Similarity computation between items or users is an important part of this approach. Multiple measures, such as Pearson correlation[77] and vector cosine[7] based similarity are used for this.

The Pearson correlation similarity of two users u_x, u_y is defined as:

$$\text{pearson_sim}(u_x, u_y) = \frac{\sum_{i \in I_{u_x u_y}} (r_{u_x, i} - \bar{r}_{u_x})(r_{u_y, i} - \bar{r}_{u_y})}{\sqrt{\sum_{i \in I_{u_x u_y}} (r_{u_x, i} - \bar{r}_{u_x})^2} \sqrt{\sum_{i \in I_{u_x u_y}} (r_{u_y, i} - \bar{r}_{u_y})^2}}$$

where I_{xy} is the set of items rated by both user \mathbf{x} and user \mathbf{y} .

The cosine-based approach defines the cosine-similarity between two users u_x and u_y as:

$$\text{cosine_sim}(u_x, u_y) = \cos(\vec{u}_x, \vec{u}_y) = \frac{\vec{u}_x \cdot \vec{u}_y}{\|\vec{u}_x\| \times \|\vec{u}_y\|} = \frac{\sum_{i \in I_{u_x u_y}} r_{u_x, i} r_{u_y, i}}{\sqrt{\sum_{i \in I_{u_x}} r_{u_x, i}^2} \sqrt{\sum_{i \in I_{u_y}} r_{u_y, i}^2}}$$

The user based top-N recommendation algorithm uses a similarity-based vector model to identify the \mathbf{k} most similar users to an active user. After the \mathbf{k} most similar users are found, their corresponding user-item matrices are aggregated to identify the set of items to be recommended. A popular method to find the similar users is the [19], which implements the nearest neighbor mechanism in linear time.

User-based approach uses similarity between users, and the items that would be recommended to a user would be based on items that another user that is most similar to them liked. A user-based recommender first finds k users whose row vectors are the most similar (and sufficiently similar) to the one of u . (k is a free parameter.) We will call these the k neighbors of u .

On the other hand, item-based approach uses the similarity between items. Then if a user likes one item, it will recommend the item that are most similar to that one. It identifies sets of items that occur together in the same shopping cart way more often than expected by chance. These sets are called frequent itemsets. To the new user who bought some of the items in a frequent itemset, some others in the same set may be recommended.

The advantages with this approach include explainability of the results, which is an

important aspect of recommendation systems, it is easy to create and use the data, etc. Some of disadvantages with this approach include decrease in performance when data gets sparse, which occurs frequently with web-related items. This hinders the scalability of this approach and creates problems with large datasets. Although it can efficiently handle new users because it relies on a data structure, adding new items becomes more complicated since that representation usually relies on a specific vector space. Adding new items requires inclusion of the new item and the re-insertion of all the elements in the structure.

3.3.1.2 Model-Based Colaborative Filtering

In this approach, models are developed using different data mining, machine learning algorithms to predict users' rating of unrated items. There are many model-based CF algorithms. Bayesian networks, clustering models, latent semantic models such as singular value decomposition, probabilistic latent semantic analysis, multiple multiplicative factor, latent Dirichlet allocation.

Latent factor models like singular value decomposition(SVD), principal component analysis(PCA), compress user-item matrix into a low-dimensional representation in terms of latent factors. One advantage of using this approach is that instead of having a high dimensional matrix containing abundant number of missing values we will be dealing with a much smaller matrix in lower-dimensional space. A reduced presentation could be utilized for either user-based or item-based neighborhood algorithms that are presented

in the previous section. There are several advantages with this paradigm. It handles the sparsity of the original matrix better than memory based ones. Non-Negative Matrix Factorization is one of the Model-based MF.

Non-Negative Matrix Factorization Non-negative matrix factorization(NMF) is a technique for obtaining low rank representation of matrices with non-negative or positive elements. Such matrices are common in a variety of applications of interest. NMF is a dimension reduction method, as the resulting decomposed matrices have a smaller number of entries than the original matrix. This means that one does not need all the entries of the original matrix to perform a decomposition, thus NMF should be able to handle missing entries in the target matrix. Indeed, factorization can be fulfilled by dropping the loss items related to the missing entries if the target loss function is a sum of per-entry losses, e.g., mean square error (MSE) or Kullback-Leibler (KL) divergence. Furthermore, the reconstructed matrix has values on entries that are missing in the original matrix. This reveals the capability of NMF for missing value imputation.

Given a data matrix A of m rows and n columns with each and every element $a_{ij} \geq 0$, NMF seeks matrices W and H of size m rows and k columns, and k rows and n columns, respectively, such that $A \approx WH$, and every element of matrices W and H is either zero or positive. The quantity k is set by the user and is required to be equal or less than the smallest of m and n . The matrix W is generally called the dictionary or basis matrix, and H is known as expansion or coefficient matrix. The underlying idea of this terminology is that a given data matrix A can be expressed in terms of summation of

k basis vectors (columns of W) multiplied by the corresponding coefficients (columns of H).

The matrices W and H are determined by minimizing the Frobenius norm:

$$||A - WH||^2$$

The minimization is carried out by some suitable iterative search process and the solution, i.e. matrices W and H, is not unique. There are many variations to the basic NMF approach wherein additional constraints, for example sparsity, are imposed on the resulting solution to limit the solution space for W and H.

Generalized Matrix Factorization Generalized Matrix Factorization (GMF) [33] is a simple nonlinear generalization of MF, which makes a prediction \hat{y}_{ui} of y_{ui} as follows:

$$\hat{y}_{ui} = h^T \phi(p_u \odot q_i)$$

where \odot denotes the element-wise product of vectors, h is a weight vector, and $\phi(\cdot)$ is an activation function. To show that MF is a special case of GMF, we can simply set $h = 1$ where 1 is the vector with all elements equal to 1. In this way, apart from the activation function, the MF model is exactly recovered by GMF.

Chapter 4

Methodology

In this section, we describe our proposed model Personalized Memory Transfer (PMT) to learn user preferences in an online conversational chit-chat setting. We use movie recommendations as a motivating example, but the proposed method is generic and can be applied to other domains as well. In this setting, there exist two types of ambiguities: 1) the user expresses preference in the form of natural language instead of explicit (ratings) or implicit (clicks/views) feedback; 2) movie mentions within the conversation may not be able to be mapped to unique item identifiers, e.g., users might simply just ask for animation movie recommendation.

We tackle the aforementioned challenges by treating each conversation as a virtual item with a combination of the (pre-trained) known item representations based on their content semantic similarity obtained from a key-value structured memory [61]. This

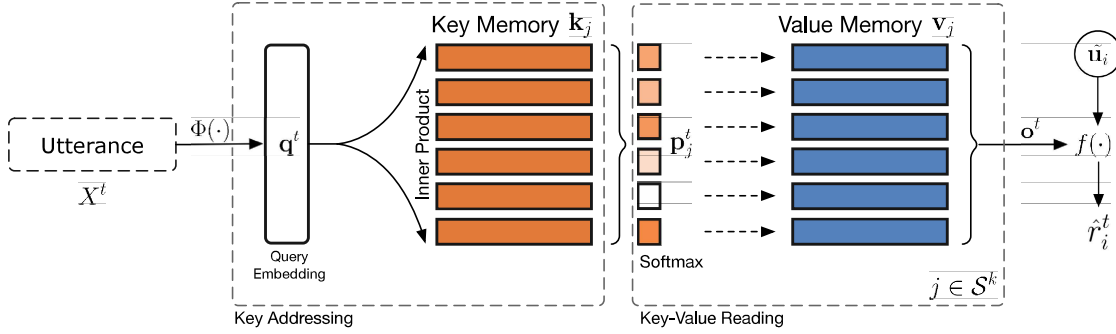


FIGURE 4.1: Illustration of PMT encoding the user’s utterance through the key-value memory structure producing the final output memory to estimate the interaction function for a new user.

virtual item can then be used as a positive instance in the pairwise training of the model.

More specifically, PMT performs online updates to a user representation throughout the conversations by consulting the stored semantic movie representation in the memory. The key-value memory structure is tailored such that the model uses the keys to address relevant memories with respect to the current user’s utterance (query) followed by a reading phase which returns the output memory using the value memory. This memory structure enables the model to transfer prior knowledge of item representations/preferences and natural language to bridge two different forms of transfer learning. The output memory returned by the key-value memory structure acts as a mixture of item preferences extracted from user’s natural language conversation, and it will guide the updates to the user representation with standard backpropagation via stochastic gradient descent. An overview of the process can be seen in Figure 4.1. The subsections below describe the architecture in detail.

4.1 Key-Value Memory

One of the major components of PMT is the key-value memory. Each movie j is represented by a memory slot with a corresponding pair of key memory \mathbf{k}_j and value memory \mathbf{v}_j . Intuitively, the key memory represents each movie in a low-dimensional semantic space to identify similar movies according to the current user’s utterance which in turn produces a relevance score weighting the appropriate value memories according to similarity. This weighted value memory or output memory acts as a weighted combination of item preferences represented in a low-dimensional latent space.

Formally, we define the key memory for the j^{th} movie via embedding a sequence of words w_l corresponding to a movie’s plot (or summary) $D_j = \{w_1, \dots, w_{|D_j|}\}$ with an encoder:

$$\mathbf{k}_j = \Phi(D_j) \tag{4.1}$$

where $\Phi(\cdot)$ is a natural language encoder or feature extractor which maps a variable length textual document to a single fixed d_e dimensional semantic vector. The encoding function could be a pre-trained language model such as Universal Sentence Encoder [9], Transformer [90], ELMo [71], and BERT [20]. Note that this can also be used to encode the user’s utterances as well. In Section 5.5, we investigate the effect of the natural language encoder on the recommendation performance by experimenting with different pre-trained sentence encoders. Once the encoder is pre-trained on a large corpus the parameters remain frozen during our training process. This is the first form

of transfer learning in PMT. We would like to point out that the definition of the keys is not restricted to a language model representation, alternative representations such as knowledge bases [61] or additional meta information such as actors, genres, or directors may also be incorporated to achieve more objective recommendations [39].

Each value memory \mathbf{v}_j is a low-dimensional latent representation of each movie j . We use the learned item representations from a pre-trained interaction function as the value memories. The pre-training can be done on a traditional large-scale collaborative filtering dataset such as MovieLens. This is the second form of transfer learning in PMT. Section 4.4 includes the details about the interaction function. Similar to the key memory, we are not restricted to setting the value memory to a specific representation from the interaction function. In addition, the flexibility of the key-value memory structure allows the key memory and value memory to be of different dimensionality.

We would like to point out that the entire memory structure is fixed and not learnable throughout the conversation. Although this work focuses on the movie domain, the representation of the keys can be easily extended to other application domains such as using product descriptions for product recommendation or paper abstracts for paper recommendation.

4.2 Key Addressing

To identify movies the user may be interested in, we query the key memory by encoding the user's utterance to a low-dimensional representation producing a relevance score for each movie. Specifically, an utterance consisting of a variable length sequence of words $X^t = \{w_1, \dots, w_{|X^t|}\}$ at a given turn t is encoded as:

$$\mathbf{q}^t = \Phi(X^t) \quad (4.2)$$

where \mathbf{q}^t represents the low-dimensional query embedding encoding the current user's utterance. Using the same natural language encoder as the key memory in the previous subsection allows the movie's plot and each utterance to be mapped into the same space permitting semantic comparison in vector space [10]. The model uses this query vector to identify relevant movies in the key memory the user may be interested in based on past utterances. Each memory slot is addressed by computing the similarity of the given utterance and each movie with:

$$s_j^t = (\mathbf{q}^t)^T \mathbf{k}_j \quad \forall j \in \mathcal{I} \quad (4.3)$$

where s_j^t expresses the user's level of interest in the j^{th} movie based on the encoded movie's summary and the t^{th} utterance \mathbf{q}^t as measured by the inner product. \mathcal{I} is the set of all movies. For example, if a user expresses interest in animated children movies, the

encoded query should produce higher relevance scores to semantically related children movies in the key memory.

4.3 Key-Value Reading

Reading the key-value memory structure bridges the two forms of transfer learning (natural language encoder and item representations/preferences) in separate semantic spaces. At a high level, the similarity computed between the query and key memories weighs the relevant value memories. In other words, the user’s utterance determines the amount of weight allocated to each movie’s value memory yielding a mixture of item representations.

Including the entire set of all movies can be computationally expensive and may introduce noise to the final memory output representation. Similar to the key-hashing performed in [61], we select a subset of top- k movies with the highest similarity score according to Eq. (4.3) which we denote as \mathcal{S}^k . To obtain the final output memory we first normalize the similarity with the softmax function:

$$p_j^t = \frac{\exp(s_j^t)}{\sum_{l \in \mathcal{S}^k} \exp(s_l^t)} \quad \forall j \in \mathcal{S}^k \quad (4.4)$$

obtaining a distribution of the current user’s preference over each movie $j \in \mathcal{S}^k$ from the utterance at turn t . Alternatively, we can also interpret this as an attention mechanism where the attention places higher weights on movies similar to the utterance.

Finally, the output memory representation is weighted by the probability each movie is relevant to the current user’s utterance in the conversation via:

$$\mathbf{o}^t = \sum_{j \in \mathcal{S}^k} p_j^t \mathbf{v}_j \quad (4.5)$$

where \mathbf{v}_j is the latent representation of movie j . This weighted value memory increases the impact of corresponding movies whose query and keys have high similarity in the semantic space while decreasing the influence of movies which may be irrelevant. Each value memory may represent concrete variations such as the genre of a movie or more subjective aspects such as visual aesthetics or a completely hidden and uninterpretable latent structure. By weighting each of the value memory according to its relevance we construct a weighted combination of preferences representing multiple degrees of each variation.

We now have the final memory output representation extracted from the user’s utterance X^t at turn t . As discussed before, we assume that it may not be possible to map user’s natural language conversation to particular movies, the memory output representation \mathbf{o}^t which is a combination from the known item representations based on their content semantic similarity obtained by the key-value memory structure serves as a good virtual “positive” item representation for the user.

4.4 Interaction Function

In this section, we first present the interaction function in the classic collaborative filtering setting and then show how we can utilize it in our proposed model via pre-training.

In collaborative filtering, the interaction function measures a user's level of interest in a given item. This interaction function can estimate the scores of unobserved entries with M users and N items in the $M \times N$ ratings matrix \mathbf{R} , which are used to rank each item. The set of all users and items are denoted as \mathcal{U} and \mathcal{I} , respectively. Formally, we assume the interaction function takes the following form which produces a ranking score for a given user $i \in \mathcal{U}$ and item $j \in \mathcal{I}$ as:

$$\hat{r}_{ij} = f(\mathbf{u}_i, \mathbf{v}_j) \quad (4.6)$$

where \mathbf{u}_i and \mathbf{v}_j are generally learnable parameters in a shared d dimensional space.

We can define multiple forms of the interaction function $f(\cdot)$. The parameters of all interaction functions can be learned by optimizing the pairwise ranking BPR loss [75] via stochastic gradient descent (SGD). The most basic interaction function is just a linear version via the inner product:

$$f(\mathbf{u}_i, \mathbf{v}_j) = \mathbf{u}_i^\top \mathbf{v}_j \quad (4.7)$$

which can yield one of the early influential algorithms in collaborative filtering: Singular Value Decomposition (SVD) [79], and other related matrix factorization (MF) based methods.

A linear function may lack the flexibility to disentangle complex item preferences and generalize to another dataset. Hence we can use a nonlinear variant generalized matrix factorization (GMF) [33]:

$$f(\mathbf{u}_i, \mathbf{v}_j) = \mathbf{h}^\top \phi(\mathbf{u}_i \odot \mathbf{v}_j) \quad (4.8)$$

where \odot is the elementwise product; $\mathbf{h} \in \mathbb{R}^d$ is an additional parameter to be learned and $\phi(\cdot)$ is a nonlinear activation function. We can adopt the rectified linear unit (ReLU) function $\phi(x) = \max(0, x)$ as the nonlinear function due to its nonsaturating behavior [66]. GMF degenerates to matrix factorization if we set the nonlinear activation function $\phi(\cdot)$ to the identity function and constrain the vector \mathbf{h} to the $\mathbb{1}$ vector with all values as 1.

In our work, we use the interaction function introduced above (either MF or GMF) as the last layer of the proposed architecture, as illustrated in Figure 4.1. We can estimate the ranking score of the output \mathbf{o}^t in Eqn.(4.5) for the user i using the interaction function as below

$$\hat{r}_i^t = f(\tilde{\mathbf{u}}_i, \mathbf{o}^t) \quad (4.9)$$

where $\tilde{\mathbf{u}}_i$ is the user representation for the utterance at turn t . As discussed in Section 4.3, \mathbf{o}^t is a combination from the known movie representations \mathbf{v}_j and can be viewed as a virtual movie representation of the utterance X^t at turn t .

Inspired by the recent success of pre-trained models in many domains, we noticed that both latent representation of movie \mathbf{v}_j and interaction function $f(\cdot)$ can be pre-trained on the existing large-scale movie rating datasets such as MovieLens. Thus, we do not have to estimate \mathbf{v}_j and \mathbf{o}^t , if there is no fine tuning. The only parameter that needs to learn is the user representation $\tilde{\mathbf{u}}_i$. During the conversation, user representation $\tilde{\mathbf{u}}_i$ is dynamically updated based on each new utterance represented by a combination of the known movies. The next section explains the loss function and the optimization procedure in model training.

4.5 Learning the User Preference Vector

Since the nature of the data resembles the implicit feedback setting, we take the pairwise assumption that a given user i prefers the observed item j^+ over the unobserved item j^- . However, without previous user interactions we cannot use standard techniques to perform the sampling. Instead, the positive item is inferred from the conversation via reading the key-value memory structure which produces a virtual item representation as \mathbf{o}^t from the utterance X^t . Since users may express positive or negative preferences towards particular recommendations throughout the conversation, we integrate a sentiment classifier $y = g(X^t) \in [0, 1]$ which handles the uncertainty associated with the negative sampling procedure. For each turn in the conversation, our training data consists of a tuple for each user utterance $X^t \in \mathcal{X}$ and sampled negative item j^- which are

used to minimize the cross-entropy objective:

$$\mathcal{L}(X^t, j^-) = -y \log(\sigma(\hat{y})) - (1 - y) \log(1 - \sigma(\hat{y})) \quad (4.10)$$

where $\sigma(x) = 1/(1 + \exp(-x))$ is the logistic sigmoid function and $\hat{y} = \hat{r}_i^t - \hat{r}_{ij^-}$.

The intuition is as follows, we want to maximize the relevance score \hat{r}_i^t of the weighted combination of movie preferences extracted by the current utterance X^t over the relevance score \hat{r}_{ij^-} from the sampled negative movie j^- assuming the sentiment is positive i.e. $y = 1$. Conversely, if the sentiment is negative the update to the user preference vector is performed in the opposite direction. In reality, utterances have varying levels of sentiment with some recommendations being favored over others, allowing y to act as a gate determining the level of satisfaction the user has with the recommendation. The only parameter learned is the new user's latent factor $\tilde{\mathbf{u}}_i$ while all other parameters are held fixed. The loss function is optimized in an online fashion using stochastic gradient descent (SGD) which allows updating the user representation as the conversation progresses. Parameter updates are performed after each seeker's utterance. Note only the utterances require the encoder function $\Phi(\cdot)$ during the online setting and the movie summaries can be encoded and cached ahead of time.

4.6 Recommendation

Since the final output memory representation acts as a virtual item extracted from the natural language conversation, it does not directly correspond to a single item and therefore cannot be used to recommend items. The final output memory is only used to evaluate the loss function with respect to the interaction function. To perform recommendations, the newly learned user representation is plugged into the interaction function. The ranking score for the new user i and item j is:

$$\hat{r}_{ij} = f(\tilde{\mathbf{u}}_i, \mathbf{v}_j) \quad (4.11)$$

Note that here we use the newly learned user representation $\tilde{\mathbf{u}}_i$ and the true item representation \mathbf{v}_j (not the estimated output memory). The top- n movies with the highest ranking scores can be presented to the user. Algorithm 1 shows the procedure on performing the recommendations in PMT. In the experiments, we evaluate these top- n recommendations against the item that the Seeker actually liked at turn t .

4.7 Response Generation

Along with the recommendation, we generate system response, using the retrieval approach [59]. In the retrieval based model, we utilize the existing data to retrieve a potential candidate for the system response.

Algorithm 1: Procedure for performing recommendations in PMT

Input: Conversation \mathcal{X} , Encoder $\Phi(\cdot)$, interaction function $f(\cdot)$, learning rate α

 Randomly initialize new user latent factor $\tilde{\mathbf{u}}_i$
for $X^t \in \mathcal{X}$ **do**

 if $Speaker == Seeker$ **then**

 Encode utterance $\mathbf{q}^t \leftarrow \Phi(X^t)$

 Compute output memory \mathbf{o}^t (Eq. (4.5))

 Compute ranking score \hat{r}_i^t (Eq. (4.9))

 Sample negative item j^-

 Update $\tilde{\mathbf{u}}_i \leftarrow \tilde{\mathbf{u}}_i - \alpha \nabla_{\tilde{\mathbf{u}}_i} \mathcal{L}(X^t, j^-)$

 else $Speaker == Recommender$

Recommend Movies (Eq. (4.11))

end
end

$$\mathbf{R}\mathbf{O} = (\mathbf{X}^t)^T \mathbf{R} \quad (4.12)$$

given the t^{th} utterance \mathbf{X}^t , and the set of response candidates, we compute the similarity of the user utterance as measured by the inner product.

We implement two simple retrieval based response generation. In the first approach we collect the system response from the training dataset as the candidate set. In this case \mathbf{R} is the set of all system responses in the training set. We rank the output candidates using the compute scores and pick the top_n . In the second approach we collect the all the user utterances from the training dataset as the candidate set. In this case \mathbf{R} is the set of all user utterances in the training set. Similar to first approach we compute the similarity of the given user utterance \mathbf{q}^t to the set of all the user utterances from train data, \mathbf{R} . For the top_n output candidates that we pick, the final response set is the immediately following response of these top_n candidates.

Chapter 5

Experiments

5.1 Datasets

We validate our proposed methodology on two public conversational datasets. The first one is called Recommendations through Dialogue (ReDial)¹ dataset [47] which consists of 206,102 utterances in 11,348 dialogues. The dialogues were crowd sourced and collected from Amazon Mechanical Turk where two users are paired up to converse around the topic of movies. Each user plays a specific role. The first user known as the seeker tries to explain their movie preferences and asks for appropriate movie suggestions. The second user known as the recommender tries to understand the seeker and provide appropriate movie recommendations. There are a total of 956 users and 51,699 movie mentions. The second dataset is called Goal-driven Recommendation

¹<https://redialdata.github.io/website/>

Dialogue (GoRecDial) which includes 170,904 utterances and 9,125 dialogues between pairs of human workers recommending movies to each other. It was collected through the task specifically designed as a cooperative game between two players working toward a quantifiable common goal. We refer the readers to [38] for full details.

Movie plots are collected from IMDB² and the interaction function is pre-trained using the latest version of MovieLens [31] consisting of 27M ratings from 283K users over 53K items. We split 80% of the ratings for training, 10% for cross-validation and the remaining 10% for hold-out purposes. Following [47] we match up the movies between the MovieLens dataset and each testbed with string matching using the authors' provided implementation³. After preprocessing and removing invalid entries we obtain 5,045 movies on the ReDial dataset and 2,065 movies on GoRecDial. We only keep movie suggestions by the recommender which was not marked as dislike by the seeker. We perform 5-fold cross-validation and report the mean. Hyperparameters are tuned on a single fold then held fixed for the remaining folds.

5.2 Evaluation Metrics

We use standard recommendation system evaluation metrics for top- n ranking, Normalized discounted cumulative gain (NDCG), Precision (P), Recall (R) and Mean reciprocal rank (MRR) [57]. Users are generally interested in only a few top-ranked movies, NDCG@ n and MRR@ n are used to compare the top- n recommendation performance.

²<https://www.imdb.com>

³<https://github.com/RaymondLi0/conversational-recommendations>

As the data resembles an implicit feedback setting where the level of user feedback is extracted from the conversation, rank aware metrics such as NDCG and MRR alone may be insufficient since a negative entry could adversely impact the metric but in reality the user may not be aware of the item’s existence. We added Recall as one additional metric for this consideration. Specifically, we treat the movies mentioned by the recommender in utterance X^t at turn t as the ground truth where the seeker did not give the recommendation ‘dislike’. Note the recommendation is performed prior to observing the utterance with the ground truth and the model has only seen the utterances prior to turn t .

In the experiments, all the movies are treated as the candidates for ranking including those the user had previously talked about. In the real conversations, a user may only refer to a movie by an inexact or vague mention and thus it would be difficult to automatically identify and exclude the mentioned movies. While both ReDial and GoRecDial datasets have annotated movie mentions, the real-world conversations may not have such information. It is worth noting that all the baseline methods also used the same set of candidate movies as our methods for a fair comparison.

5.3 Baselines and Settings

We validate the effectiveness of our model against a few baseline methods.

Random: The top- n recommendations are randomly selected from all the candidate movies in a uniform distribution.

Popularity: The movies with the highest popularity that occur in the MovieLens training set are presented to the user. While simple, this baseline could sometimes yield competitive performance in the benchmark recommendation tasks [18].

TF-IDF (Term frequency inverse document frequency): The movies are ranked according to the TF-IDF vector space similarity between the user’s utterance and each movie’s plot via the inner product.

Semantic Retrieval: This method uses Eq. (4.3) to rank the movies according to the inner product similarity where the encoder is the Universal Sentence Encoder. Movies with the highest similarity to the utterance are presented to the user.

SVD (Singular Value Decomposition) [79] and **NMF** (Non-negative Matrix Factorization) [26]: These two baselines are classic collaborative filtering methods based on matrix factorization (Section 3.3.1.2). We used movie preferences from the conversations to form user rating matrices (binary: like vs dislike) and then learn latent user and item factors for rating predictions. These methods require movie mentions identifiers annotated in each utterance.

ConvAE: This Conversational Autoencoder (ConvAE) approach was proposed in [47] (together with the ReDial dataset). It consists of a dialogue generation model based

on hierarchical recurrent encoder-decoder, a sentiment prediction model, and an auto-encoder recommender. Similar to SVD and NMF, this approach requires explicit movie annotations in the conversations.

DropoutNet [91]: A deep learning based approach which adapts a deep neural network to handle the user cold-start problem on top of weighted matrix factorization (WMF), which demonstrated the state-of-the-art performance on public benchmarks.

For PMT, we first pre-train the two interaction functions Matrix Factorization (MF) and Generalized Matrix Factorization (GMF) with movie representations on the MovieLens dataset. All hyperparameters are obtained from a grid search by performance on the held out cross-validation set. We searched for the best latent dimension or embedding size from $\{10, 20, 30, 40\}$, L_2 between 0.1 to $1e-7$, and the model was optimized with the SGD variant Adam [41] using a standard learning rate of 0.001. We applied the pre-trained Universal Sentence Encoder with Transformer (USE-Transformer) [9] to encoder natural language texts in all the experiments (unless specified otherwise). Section 5.5 studies the effect of the sentence encoder on the recommendation performance.

In the experiments, the sentiment classifier was a logistic regression model with the USE-Transformer encoding of a given utterance as input. To train a sentiment classifier for each dataset, we used the labeled sentiment data from each dataset together with the labeled data from the Movie Review dataset⁴, which resulted in 218,809 training instances for Redial and 75,005 for Goredcdial. The movie review dataset is in the same domain of movie discussions and shares some similar characteristics with the two

⁴<https://ai.stanford.edu/~amaas/data/sentiment/>

testbeds. It has limited but high-quality binary sentiment labels (25,000 training examples). The majority of training data for sentiment classifiers still came from ReDial and GoRecDial themselves.

It is worth noting that several parts of the proposed model is pre-trained including sentence encoder/embeddings and sentiment classifier (pre-trained on sentiment labels), as well as the interaction function $f(\cdot)$ in Eqn.(4.9) and the latent representation of movie \mathbf{v}_j (pre-trained on MovieLens). Once they are pre-trained, the parameters are held fixed and not fine-tuned to the downstream recommendation task as we considered the fact that the labeled domain data is limited. As demonstrated in Section 5.4, even without fine-tuning of the pre-trained models, the proposed approach demonstrated good improvement over the baseline methods. As shown in Algorithm 1, each new user latent factor $\tilde{\mathbf{u}}_i$ is the only learnable parameter during training. It is randomly initialized from a uniform distribution with the range computed from the variance of the pre-trained user representations maintaining the relative scale with respect to other parameters. During the conversation we use SGD with a learning rate of 0.1 to optimize the new user representation. At each turn t we update the parameters $\tilde{\mathbf{u}}_i$ and randomly sample negative items from the k least similar movies. All the experiments were conducted on a server with 2 Intel E5-2630 CPUs and 4 GeForce GTX TITAN X GPUs. The proposed models were implemented in PyTorch with the source code publicly available at GitHub⁵.

⁵<https://github.com/agodavarthy/PMT>

	P@1	P@3	R@1	R@3	R@10	R@25
Random	0.0001	0.0002	0.0000	0.0004	0.0019	0.0048
Popularity	0.0035	0.0020	0.0024	0.0048	0.0112	0.0220
TF-IDF	0.0000	0.0000	0.0000	0.0000	0.0000	0.0080
Semantic	0.0048	0.0036	0.0037	0.0086	0.0214	0.0411
SVD	0.0000	0.0022	0.0000	0.0026	0.0116	0.0317
NMF	0.0000	0.0002	0.0000	0.0016	0.0095	0.0274
ConvAE	0.0014	0.0024	0.0014	0.0072	0.0408	0.1013
DropoutNet	0.0039	0.0056	0.0031	0.0136	0.0346	0.0577
PMT-MF	0.0017	0.0019	0.0017	0.0058	0.0153	0.0336
PMT-GMF	0.0060†	0.0061	0.0046†	0.0144	0.0414†	0.0798
	NDCG@3	NDCG@10	NDCG@25	MRR@3	MRR@10	MRR@25
Random	0.0003	0.0009	0.0016	0.0003	0.0006	0.0009
Popularity	0.0042	0.0065	0.0092	0.0048	0.0058	0.0067
TF-IDF	0.0000	0.0000	0.0018	0.0000	0.0000	0.0003
Semantic	0.0069	0.0117	0.0166	0.0068	0.0098	0.0113
SVD	0.0014	0.0046	0.0094	0.0006	0.0024	0.0036
NMF	0.0008	0.0036	0.0079	0.0002	0.0018	0.0029
ConvAE	0.0043	0.0157	0.0302	0.0034	0.0084	0.0118
DropoutNet	0.0096	0.0173	0.0232	0.0074	0.0131	0.0151
PMT-MF	0.0042	0.0077	0.0123	0.0034	0.0058	0.0072
PMT-GMF	0.0108†	0.0208†	0.0306	0.0091†	0.0160†	0.0192†

TABLE 5.1: Experimental results on the ReDial dataset for different methods reporting Precision (P), Recall (R), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) for different cut offs. The best results are highlighted in bold. † denotes the improvement over the best result of the baselines is statistically significant based on the paired t-test (p-value < 0.05).

5.4 Baseline Comparison

In this section, we present the results of our proposed Personalized Memory Transfer (PMT) approach against the baseline methods on the two testbeds. Table 5.1 shows the results on the ReDial dataset. As we can see, the Random baseline yielded very small values in all the metrics, indicating the difficulty of this recommendation task. The Popularity baseline performs much better than Random but generally worse than the other more competitive methods due to its lack of any form of personalization. The

Semantic Retrieval method yielded much better results than TF-IDF, which is likely due to the fact that the user conversations often do not contain the keywords in the movie plots. There often exist some vocabulary gap between utterances and movie plots. For example, in the example dialogue shown in Figure 1.1, the seeker was looking for a “suspenseful” movie. The plot of a relevant movie *A Space Odyssey* does not include “suspenseful” but it contains a closely related word “mysterious”. The much improved results of the Semantic baseline over TF-IDF demonstrated the advantages of the semantic sentence encoder in relating user utterances with movie plots.

The matrix factorization methods SVD and NMF performed better than Random, Popularity, and TF-IDF but worse than Semantic Retrieval, which indicates the importance of utilizing natural language information for conversational recommendations as SVD and NMF only looked at annotated movie preferences. ConvAE did not perform well on the very top ranked results as evidenced by the metrics of P@1, P@3, R@1, R@3, and NDCG@3, while obtaining the best result in R@25. It is worth noting that ConvAE relies on annotated movie mentions in the utterances, which may not be available in the real world scenarios. Nonetheless, the proposed PMT-GMF model yielded the best results in all the metrics except R@25, without utilizing the information of annotated movies. The majority of the improvements of PMT-GMF over the best baseline results were statistically significant based on the paired t-test ($p\text{-value} < 0.05$). DropoutNet generated the second best results in most metrics, which indicated the effectiveness of nonlinearity modeling on ReDial. This nonlinearity effect can also be seen via the

	P@1	P@3	R@1	R@3	R@10	R@25
Random	0.0005	0.0003	0.0005	0.0011	0.0042	0.0110
Popularity	0.0039	0.0014	0.0039	0.0044	0.0110	0.0368
TF-IDF	0.0028	0.0021	0.0028	0.0063	0.0126	0.0217
Semantic	0.0089	0.0068	0.0089	0.0204	0.0461	0.0792
SVD	0.0000	0.0028	0.0000	0.0083	0.0488	0.1432
NMF	0.0000	0.0004	0.0000	0.0032	0.0238	0.0680
ConvAE	0.0068	0.0081	0.0137	0.0325	0.1220	0.2434
DropoutNet	0.0064	0.0087	0.0064	0.0262	0.0650	0.1380
PMT-MF	0.0169	0.0145	0.0169	0.0437	0.1007	0.1825
PMT-GMF	0.0202†	0.0176†	0.0202†	0.0528†	0.1231	0.2169
	NDCG@3	NDCG@10	NDCG@25	MRR@3	MRR@10	MRR@25
Random	0.0009	0.0019	0.0036	0.0007	0.0012	0.0017
Popularity	0.0042	0.0063	0.0126	0.0041	0.0048	0.0066
TF-IDF	0.0047	0.0070	0.0092	0.0036	0.0052	0.0058
Semantic	0.0154	0.0225	0.0325	0.0117	0.0176	0.0198
SVD	0.0047	0.0186	0.0412	0.0020	0.0093	0.0151
NMF	0.0017	0.0087	0.0194	0.0004	0.0040	0.0069
ConvAE	0.0239	0.0552	0.0849	0.0210	0.0354	0.0431
DropoutNet	0.0177	0.0314	0.0489	0.0118	0.0206	0.0254
PMT-MF	0.0324	0.0523	0.0720	0.0248	0.0370	0.0425
PMT-GMF	0.0387†	0.0634†	0.0861	0.0286†	0.0445†	0.0509†

TABLE 5.2: Experimental results on the GoRecDial dataset for different methods reporting Precision (P), Recall (R), normalized discounted cumulative gain (NDCG) and mean reciprocal rank (MRR) for different cut offs. The best results are highlighted in bold. † denotes the improvement over the best result of the baselines is statistically significant based on the paired t-test (p-value < 0.05).

comparison between the results of PMT-MF and PMT-GMF. PMT-GMF obtains better performance than the linear counterpart PMT-MF which suggests the presence of more complex nonlinear interactions may be required to disentangle user preferences and transfer the knowledge to another dataset.

On the GoRecDial dataset as shown in Table 5.2, the results show a similar pattern with the ones in Table 5.1 on ReDial, but with larger improvements observed on PMT-MF and PMT-GMF over the baseline methods. PMT-MF yielded the second best in the majority

of the metrics after PMT-GMF. The PMT-GMF model obtained better results than all the baselines across all the metrics except R@25, which reaffirms the effectiveness of the proposed model. Similar to the results on ReDial, PMT-GMF outperforms PMT-MF, which demonstrates the advantage of modeling nonlinearity via the interaction function.

5.5 Effect of Sentence Encoder

In this section, we investigate the effect of the underlying sentence encoder on the recommendation performance of the proposed model. Specifically, We experiment with four popular deep contextualized pre-trained sentence encoders to transform the utterances in the conversations into sentence embeddings. The encoders are as follows:

- ELMo [71]. It uses a bi-directional LSTM to compute contextualized character-based word representations. We used TensorFlow Hub implementation of ELMo⁶, trained on the 1 Billion Word Benchmark.
- BERT [20]. It is a deep embedding model that learns vector representations of words and sentences by training a deep bidirectional Transformer network. We used the uncased BERT-Base model⁷ trained on English Wikipedia and BooksCorpus.
- Universal Sentence Encoder (USE) [9]. It has two variants: one is trained with a Deep Averaging Network (USE-DAN) and another with a Transformer network

⁶<https://tfhub.dev/google/elmo/2>

⁷<https://github.com/google-research/bert>

(USE-Transformer). We used TensorFlow Hub implementation of USE⁸, trained on a variety of data sources including Wikipedia, web news, web question-answer pages, and supervised data from the Stanford Natural Language Inference (SNLI) corpus.

	P@1	P@3	R@1	R@3	R@10	R@25
ELMo	0.0039	0.0040	0.0030	0.0094	0.0266	0.0528
BERT	0.0056	0.0057	0.0044	0.0138	0.0368	0.0709
USE-DAN	0.0035	0.0037	0.0027	0.0089	0.0254	0.0509
USE-Transformer	0.0060	0.0061	0.0047	0.0146	0.0414	0.0798
	NDCG@3	NDCG@10	NDCG@25	MRR@3	MRR@10	MRR@25
ELMo	0.0070	0.0134	0.0201	0.0060	0.0104	0.0125
BERT	0.0103	0.0189	0.0275	0.0087	0.0147	0.0174
USE-DAN	0.0066	0.0127	0.0192	0.0056	0.0097	0.0117
USE-Transformer	0.0108	0.0208	0.0307	0.0091	0.0161	0.0192

TABLE 5.3: Experimental results of PMT-GMF on the ReDial dataset with different sentence encoders. The best results are highlighted in bold.

Table 5.3 and Table 5.4 contain the experimental results of the proposed PMT-GMF model with different sentence encoders on the two datasets respectively. We can see the USE-Transformer encoder achieved the best results in all the metrics on both datasets. The BERT encoder yielded very similar results with USE-Transformer on ReDial across all the metrics. On GoRecDial, both variants of USE obtained slightly better results than BERT. The minor improvement of USE-Transformer over BERT could come from the fact that USE-Transformer used more diverse sources of data (which include some informal texts) for pre-training than BERT did. The results were consistent with some existing work in other recommendation domains which showed USE could generate

⁸<https://tfhub.dev/google/universal-sentence-encoder/4>

	P@1	P@3	R@1	R@3	R@10	R@25
ELMo	0.0146	0.0127	0.0146	0.0381	0.0900	0.1617
BERT	0.0169	0.0145	0.0169	0.0436	0.1007	0.1825
USE-DAN	0.0189	0.0167	0.0189	0.0503	0.1181	0.2093
USE-Transformer	0.0202	0.0176	0.0202	0.0528	0.1231	0.2169
	NDCG@3	NDCG@10	NDCG@25	MRR@3	MRR@10	MRR@25
ELMo	0.0279	0.0461	0.0634	0.0206	0.0322	0.0371
BERT	0.0324	0.0523	0.0720	0.0248	0.0370	0.0425
USE-DAN	0.0367	0.0605	0.0826	0.0271	0.0424	0.0486
USE-Transformer	0.0387	0.0634	0.0861	0.0286	0.0445	0.0509

TABLE 5.4: Experimental results of PMT-GMF on the GoRecDial dataset with different sentence encoders. The best results are highlighted in bold.

marginally better recommendation results than BERT as a sentence encoder [32]. Overall, the four state-of-the-art pre-trained text encoders yielded results in a comparable range.

5.6 Effect of Embedding Size

In this section, we analyze the effect of embedding size of key embedding \mathbf{k}_j (Eqn.(1)), query embedding \mathbf{q}^t (Eqn.(2)), movie embedding \mathbf{v}_j (Eqn.(5)), and user embedding \mathbf{u}_i (Eqn.(7)), on the performance of our model. Figure 5.1 presents the performance comparison with respect to the length of embedding varied from 10 to 40 on the two datasets in different metrics. As we can see, on the GoRecDial dataset the metrics in Recall and NDCG first increase and then decrease, while on the ReDial dataset the performance in all the metrics improves when the embedding size increases. This may be due to the fact that the GoRecDial dataset has less dialogues and utterances and thus the model is more likely to overfit when the embedding size (or model complexity)

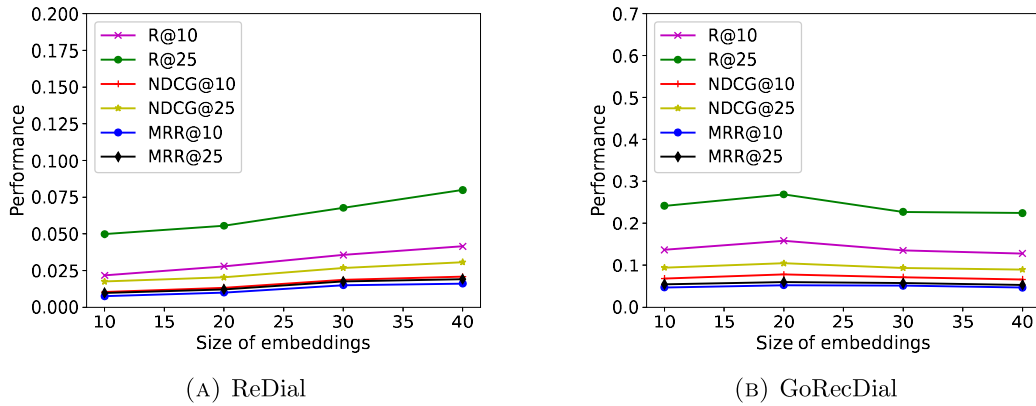


FIGURE 5.1: Effect of embedding size on the proposed PMT-GMF model on the two datasets in different metrics

increases. In general, the results across different metrics have a consistent pattern over the embedding size.

5.7 Effect of Conversation Length

In this section, we examine the effect of the length of the conversation to determine whether updates to the user representation leads to improved performance. We bin the results of each recommendation based on the number of turns in a conversation. For example, if the model has seen 3 turns of conversation before a recommendation, the result of that recommendation will be categorized into the 3-turn bin. Figure 5.2 plots recall at 25 ($R@25$) varying the number of turns from 1 to 5. We do not report the other metrics since they show similar trends.

As we can see, the Random and Popularity baselines seem insensitive to the number of

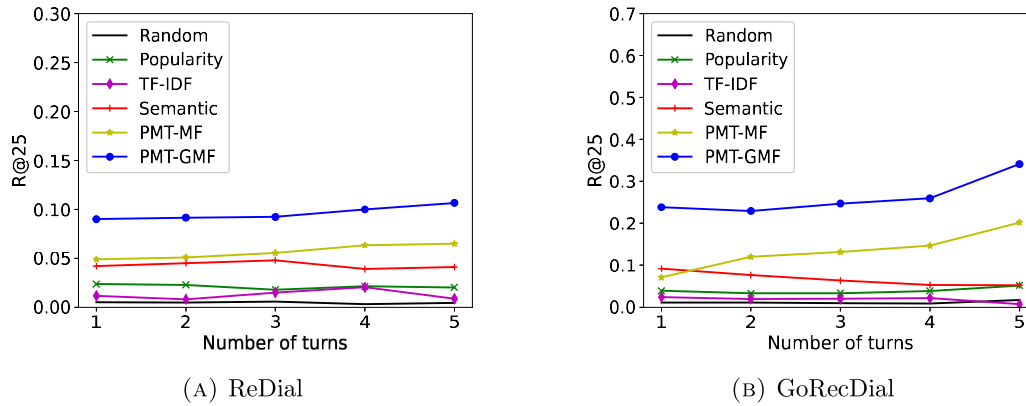


FIGURE 5.2: Recall at 25 ($R@25$) of different methods over the number of turns in the conversations.

turns in a conversation, which is expected since they do not update the recommendations based on the conversations. The performance of TF-IDF seems to peak around 4 turns on both datasets potentially due to the dependency on keywords and the increasing complexity of the user's utterances as the conversation progresses. The Semantic baseline shows comparable results with TF-IDF on ReDial and noticeably better performance on GoRecDial. On the other hand, PMT-GMF and PMT-MF demonstrate a steady increase and upward trend in performance as the number of turns increase on both datasets. The pattern is more visible on the GoRecDial dataset for the two proposed models. These results illustrate the effectiveness of the proposed models in updating the user representation over time and learning better user preferences. For all the different numbers of turns, PMT-GMF yields much better results than the other methods. In general, PMT-MF delivers the second best performance on the two testbeds and is followed by the Semantic baseline. The pattern in different number of turns is generally consistent with the overall results shown in Table 5.1 and Table 5.2.

5.8 Effect of Number of Layers for Different Embedding Sizes

In this section, we analyze the effect of number of layers on the performance of our model. Figures 5.3-5.6 present the performance comparison with respect to the number of layers ranging from 2 to 5 for each of the embedding size from 10 to 40 on the two datasets for Recall. As we can see from the figures, for both Redial and GoRecDial datasets, R@25 performs the best. For Redial dataset, for embedding size of 10, the best performance is when the number of layers are 3, however for the embedding size of 20, the best performance is for network with 2 layers. As the embedding size increases, it seems that the networks with more number of layers is performing better. This shows that the larger embedding size has higher representational capacity which in turn is giving good performance.

For the GoRecdial dataset, in general, as the embedding size in increasing from 10 to 30, the networks with more number of layers seem to perform better, although for embedding size of 10, network with 5 layers is equally good as network with 2 layers. As the embedding size increases to 40, however the best performance is for 2 layered network, suggesting it might be overfitting. We believe the difference in performance between Redial and GoRecDial data might be due to the richness of the redial data with longer sentences in the conversations. While a significant amount of GoRecDial data has single words like ACCEPT/REJECT.

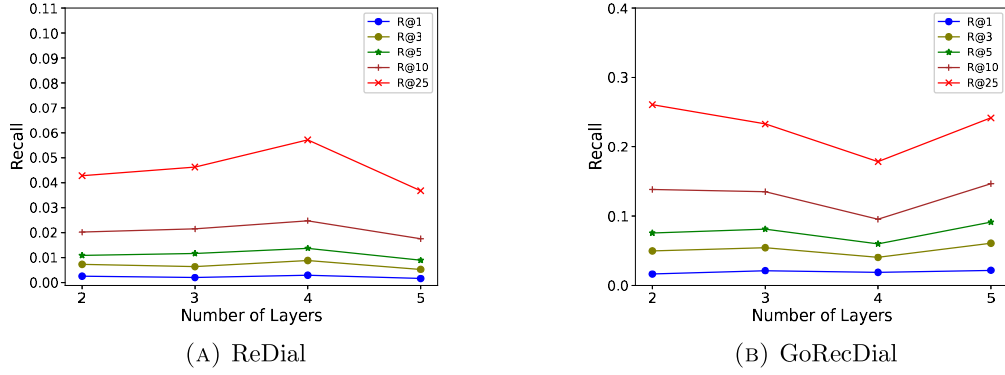


FIGURE 5.3: Effect of number of layers with embedding size 10 on the proposed PMT-GMF model on the two datasets for Recall

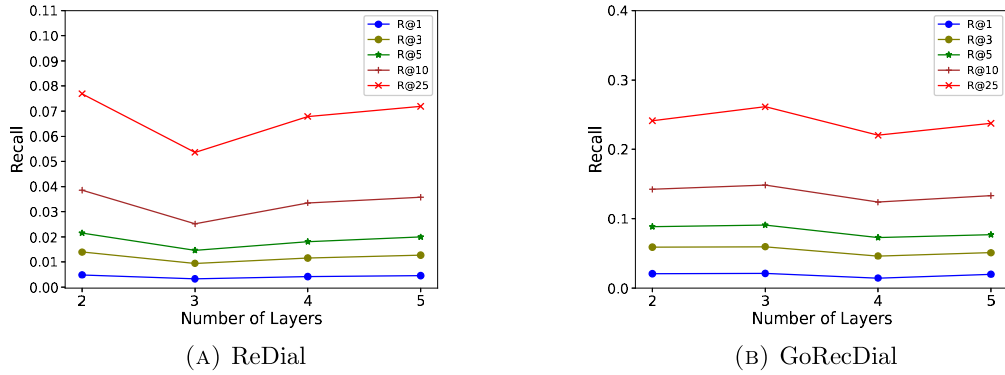


FIGURE 5.4: Effect of number of layers with embedding size 20 on the proposed PMT-GMF model on the two datasets for Recall

5.9 Efficiency Analysis

In this section, we empirically study the efficiency of the proposed PMT-GMF model. Specifically, we tested how fast on average the model can make recommendations at run time given an utterance in the conversation. We recorded the response time for each instance in the test set. The response time can be further decomposed into two parts: time for encoding an utterance and time for generating a ranked list of recommended movies. The experiments were conducted on one GeForce GTX TITAN X GPU. Table

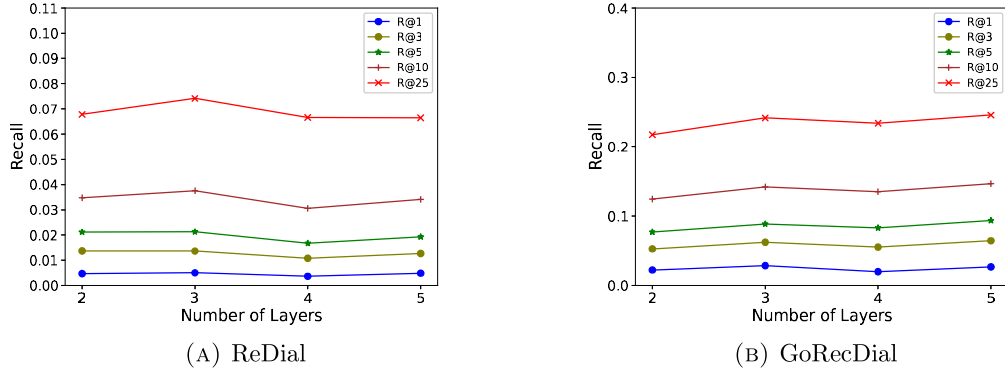


FIGURE 5.5: Effect of number of layers with embedding size 30 on the proposed PMT-GMF model on the two datasets for Recall

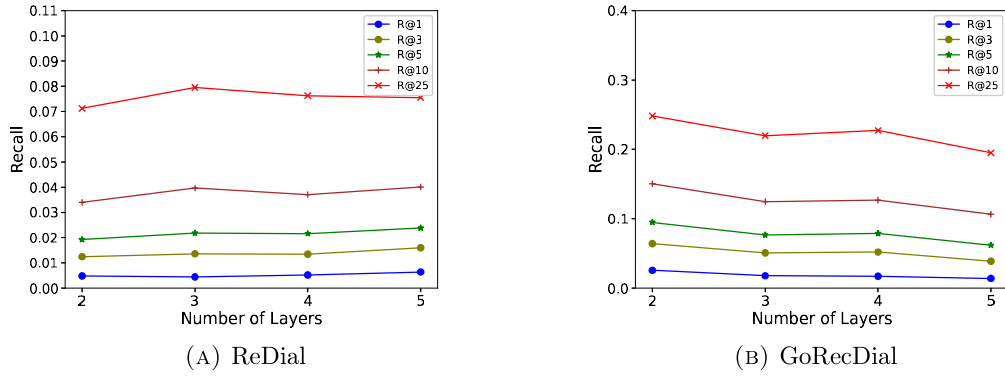


FIGURE 5.6: Effect of number of layers with embedding size 40 on the proposed PMT-GMF model on the two datasets for Recall

5.5 contains the average response time (in seconds) of PMT-GMT with two different sentence encoders on the two datasets. As we can see, the encoding process constitutes a large fraction of the response time on both datasets, since BERT and USE-Transformer are large language models with many parameters. The encoding took slightly longer time on GoRecDial than on Redial probably because the average length of utterances on Redial is a bit larger. On the other hand, the ranking time on Redial was slightly longer than that on GoRecDial, which is likely due to the fact that Redial contains

PMT-GMF	Redial			GoRecDial		
	Encoding	Ranking	Total	Encoding	Ranking	Total
BERT	0.6490	0.1106	0.7596	0.6973	0.0960	0.7935
USE-Transformer	0.6040	0.1012	0.7163	0.6189	0.0898	0.7087

TABLE 5.5: Average response time (in seconds) of PMT-GMT with two different sentence encoders on the two datasets: BERT and USE-Transformer.

more candidates movies than GoRecDial does. Overall, the total response times on both datasets with different encoders were in a similar range and less than 0.8 second. It is worth noting that there is abundant room to further improve the efficiency by using Knowledge distillation, e.g., DistilBERT [81], to compress large sentence encoders. We will further investigate this in the future work.

5.10 Qualitative Study

In this section, we conduct a qualitative study on a specific conversation instance and see how PMT-GMF learns user preferences as the dialogue progresses. Figure 5.7 shows an example conversation from the GoRecDial dataset between recommendation seeker (Seeker) and recommender (Rec). Marked in blue in the figure are the top 10 movie recommendations by PMT-GMF up to the conversation at that turn (based on the ranking score in Eqn.(11)). The positive recommendation based on the ground truth is marked in bold.

At the beginning of the conversation, the Seeker expressed his/her interest in “computer animate comedies”, “fantasy movies”, and “Harry Potter”. As we can see from the first

Seeker: My favorite movies are computer animate comedies. I also like some fantasy movies like Harry Potter.

Rec: Do you like any family movies?

Seeker: Sure, I like all Walt Disney movies such as Meet the Robinsons

Rec: How about the family movies where there is a princess in it?

Seeker: I love the Toy Story movies

Rec: How about any of the princess movies?

Seeker: I don't know if Harry Potter has a princess in it, but that would be the only one

Model: *Forrest Gump* | ***Shrek*** | *Monsters, Inc.* | *Finding Nemo* | *Harry Potter and the Prisoner of Azkaban* | *Harry Potter and the Chamber of Secrets* | *American Beauty* | *Chicken Run* | *Up* | *Groundhog Day*

Rec: RECOMMEND Shrek

Seeker: Why did you recommend this movie?

Rec: It was a computer animated comedy and also similar to the movies they said they liked.

Seeker: ACCEPT Shrek

Rec: Why did you accept the movie?

Seeker: I like computer animated movies

Model: *Shrek* | *Monsters, Inc.* | ***Finding Nemo*** | *Harry Potter and the Prisoner of Azkaban* | *Forrest Gump* | *Harry Potter and the Chamber of Secrets* | *American Beauty* | *Up* | *Ratatouille* | *Harry Potter and the Half-Blood Prince*

Rec: RECOMMEND Finding Nemo

Rec: have you seen Finding Nemo?

Seeker: no, but I would love that too

FIGURE 5.7: an example conversation from the GoRecDial dataset between recommendation seeker (Seeker) and recommender (Rec). Marked in blue in the figure are the top 10 movie recommendations by PMT-GMF. The positive recommendation is marked in bold based on the ground truth.

top 10 recommendations by the proposed model, two movies belong to Harry Potter series. The “computer animate comedies” type of movies include *Shrek*, *Monsters, Inc.*, *Finding Nemo*, *American Beauty*, *Chicken Run*, and *Up*. These results demonstrate that the proposed approach can learn user preferences even when they are expressed at a high level (e.g., “computer animate comedies”) with inexact movie mentions (e.g., “Harry Potter”). Only 2 out of the 10 recommendations seemed to be what the Seeker was looking for. *Forrest Gump* showed up as the top result probably because it is a

comedy and a very popular movie. *Groundhog Day* is a fantasy comedy film but not computer animated.

The recommender (Rec) in the dataset recommended *Shrek* which was accepted by Seeker. It is worth noting that only *Shrek* is considered as a positive instance for this recommendation in the evaluation while some other top ranked movies by our model could also be relevant in the real world if they are presented to the Seeker. This offline evaluation methodology may explain why all the experimental results have relatively low values.

After the Seeker accepted the first recommendation, he or she further indicated “I like computer animated movies”. As we can see, all the top three movies ranked by our model belong to “computer animated movies”. The positive instance *Finding Nemo* appeared at the 3rd position. *Forrest Gump* was pushed down and *Groundhog Day* did not appear in the top 10 results any more. *Up* was moved up from the previous position. A new movie *Ratatouille* emerged in the top recommendations, which is a computer-animated comedy and released by Walt Disney as what was wanted by the Seeker. These results demonstrate that our model could effectively update preferences and recommendations based on the new utterances.

5.11 Response Generation

In this section, we report the response generation module implemented using retrieval-based [58]. We implement two approaches for response generation. In the first approach(Rec1), we pool the all the responses of the recommender from the training dataset.

Given a seekers utterance, we compute the semantic similarity between the representation of the seeker utterance and the pool of the recommender responses using dot product. We then rank them in the order of their similarity scores. In the second approach(Rec2) [59], we build the candidate set by pooling all the seeker utterances from the training dataset. We compute the semantic similarity between the seeker utterances and the candidate set, that is built using the seeker utterances. The set of recommender responses are the immediately following response of the candidates that are ranked higher in the similarity scores.

In the Figures 5.8, 5.9, we present the example of response generation using both the approaches for Redial dataset. In the first set of examples in Figure 5.8, the seeker is looking for a comedy movies, the Rec1 retrieves one relevant candidate in top 5 results, whereas Rec2 has two relevant candidates in the top 5 results. In the second set of examples in Figure 5.9, the seeker3 is looking for comedy movie. The recommendations done by Rec1 include one relevant result, *The Boss Baby*, whereas Rec2 retrieves one romantic comedy(*Wedding Singer*), and two comedies(*Grown Ups*, *Mrs.Doubtfire*). The seeker4 in Figure 5.9 is seeking a romantic comedy. *Toy Story* recommended by Rec1

Seeker1:	<i>No , I think I would like to watch Easy A . It's funny</i>
Rec1:	1) I would also recommend Green Lantern if you like Ryan Reynolds or Deadpool 2 . 2) He is always too much haha. What about The Heat with Sandra Bullock and Melissa McCarthy. 3) Hi there, how can I help you with today? 4) What kind of movies do you like? 5) I love them . I think Tron was a musical
Rec2:	1) sure :) . Oh well I think you will like Get Out Another movie I'd recommend is Moneyball , Brad Pitt's in it . It's about baseball and economics 2) I watched Fist Fight last week. It was really funny. 3) I also love Paul Blart: Mall Cop and Paul Blart: Mall Cop 2 4) Yeah, it was intense. I would definitely check it out! 5) Spy finds me interesting

Seeker2:	<i>I also like Big Daddy and The Waterboy.</i>
Rec1:	1) Split was pretty scary to me as well. 2) Yup. 3) Hey 4) Have you seen Madea Goes to Jail . It was funny 5) texas high schools; what about a high school movie? there is some romancing in American Pie
Rec2:	1) Raiders of the Lost Ark was good 2) tell me a movie you like or a genre you are interested in? . Cool 3) I also liked My Best Friend's Wedding . Used to watch that one a lot . 4) Oh and The Wedding Singer . what are some you enjoy? . okay, great 5) Great! Have you ever seen How to Lose a Guy in 10 Days ?

FIGURE 5.8: Examples of Response Generation for a given Seeker dialog for Redial dataset.

meets the criteria of comedy to a certain level. A couple of recommendations done by Rec2 *Barbershop: The Next Cut* It, *Notting Hill* fall under the category of comedy and romantic comedy respectively. In general Rec2 which is based on computing the similarity between the user utterance seems to be performing slightly better than the Rec1, which relies on the similarity score between the user/seeker utterance and system

Seeker3:	<i>I have seen that movie and it is definitely hilarious. I have yet to watch the next one though, 22 Jump Street .</i>
Rec1:	<ol style="list-style-type: none"> 1) Hello! 2) I'm ok , Have you seen The Lion King , The Boss Baby ? 3) Yes, it has Hilary Swank, Josh Hartnett and Scarlett Johansson 4) Are you looking for any genre specifically? 5) Also Me Before You .
Rec2:	<ol style="list-style-type: none"> 1) maybe The Wedding Singer ? . I also thought Grown Ups was cute 2) We are watching Mrs. Doubtfire right now. Have you seen this one yet? 3) Honestly I haven't heard much about it. I need to watch the trailer for it. 4) Excellent, I hear you're looking for some movies? 5) Yes you would enjoy Winchester or Insidious: Chapter 4

Seeker4:	<i>I am great. . It's a nice day. . I also really like The Holiday . . I like Shallow Hal too. . I hope you can help by just suggesting one movie.</i>
Rec1:	<ol style="list-style-type: none"> 1) That was made with real teenagers and is pretty good. It's about a school shooting. 2) The Untouchables 3) Very funny movie! What about Toy Story ? 4) The Blind Side is another good one with Sandra Bullock 5) Power Rangers is also nice
Rec2:	<ol style="list-style-type: none"> 1) Have you had a chance to see Barbershop: The Next Cut It was cute and it's fairly new. 2) I liked Notting Hill 3) I havent seen that one, but have you seen Shutter Island ? Thats a pretty good Mystery movie 4) have you seen the The Ritual ? . british movie about friends going hiking 5) Have you ever seen Beauty and the Beast with Emma Watson? . It's so magical.

FIGURE 5.9: Examples of Response Generation for a given Seeker dialog for Redial dataset.

response.

In Figures 5.10,5.11, we present the results of the response generation for Gorecdial dataset. In Figure 5.10, seeker1 is looking for a thriller/crime movie. Rec1 has three

Seeker1:	<i>Hello, I like crime-dramas and thrillers</i>
Rec1:	<ol style="list-style-type: none"> 1) do you like alfred hitchcock movies? 2) Why did you accept the movie? 3) RECOMMEND Jurassic Park 4) Why did you accept the movie? 5) I recommended this because they said they like crime movies, and my crystal ball had one
Rec2:	<ol style="list-style-type: none"> 1) It's an American war movie I think you would enjoy. 2) RECOMMEND Watchmen 3) RECOMMEND Full Metal Jacket 4) RECOMMEND Miss Congeniality 5) RECOMMEND Airplane!
Seeker2:	<i>Hello, I am looking for A movie recommendation. I tend to like popular, well known movies.</i>
Rec1:	<ol style="list-style-type: none"> 1) Why did you accept the movie? 2) Do you like coming of age movies? 3) RECOMMEND American Beauty 4) Why did you accept the movie? 5) RECOMMEND Taxi Driver
Rec2:	<ol style="list-style-type: none"> 1) Is there a time frame that you like? For instance, do you like newer movies, or 80s movies, or classics? 2) i've got two to recommend, let's see if i can guess right 3) What directors do you like? 4) It's an American war movie I think you would enjoy. 5) RECOMMEND Minority Report

FIGURE 5.10: Examples of Response Generation for a given Seeker dialog for GoRec-dial dataset.

relevant retrievals in top five results, including *alfred hitchcock*, *Jurassic Park* and mention of the word *crime* in one of the results. Only one output among the top five results for Rec2 is slightly close to the criteria that the user is seeking, i.e., *watchmen*. In Figure 5.11, seeker4 is looking for comedy/crime/blackandwhite. While none of the to five retrievals of Rec1 are relevant, just one retrieval among to five for Rec2 is close to the seeker's criteria. Although *Lawrence of Arabia* is not a black and white movie, it is historical drama, which falls in the category of black and white classics.

Seeker3:	<i>Foreign films are a bit out of my comfort zone. War films can be okay. Tarantino is okay too but not necessary</i>
Rec1:	<ol style="list-style-type: none"> 1) RECOMMEND Pulp Fiction 2) seeker likes this genre 3) It is a comedy film, the category suggested. 4) RECOMMEND Fight Club 5) do you like historical drama?
Rec2:	<ol style="list-style-type: none"> 1) RECOMMEND Iron Man 3 2) RECOMMEND O Brother, Where Art Thou? 3) RECOMMEND Children of Men 4) Do you prefer a little horror or a little drama with your comedy? 5) Are you into mystery?
Seeker4:	<i>comedy or black and white I also like crime</i>
Rec1:	<ol style="list-style-type: none"> 1) RECOMMEND Full Metal Jacket 2) Why did you accept the movie? 3) RECOMMEND High Fidelity 4) RECOMMEND Aspen Extreme 5) What movies do you like?
Rec2:	<ol style="list-style-type: none"> 1) RECOMMEND Sling Blade 2) RECOMMEND Lawrence of Arabia 3) what genres do youlike? 4) I sure can, can you tell me some movie genres you enjoy to start with? 5) It's an American war movie I think you would enjoy.

FIGURE 5.11: Examples of Response Generation for a given Seeker dialog for GoRec-dial dataset.

In general, Rec2 performs slightly better than Rec2 for Redial data and for GorecDial data, the performance of both the recommenders seem to be similar. We believe it is because, the seeker utterances in Redial dataset have longer sentences, it can semantically find similarity with other seeker utterances. However for the Gorecdial dataset, most of the seeker utterances are short,so Rec2 could not find many semantically similar seeker utterances. Rec1 performed much better for Gorecdial dataset, which compared the seeker utterance with the system recommendation directly.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

To summarize, in this work we propose PMT to learn representation for a new user in an online fashion in a conversation setting. On the one hand, our work does not depend on historical interaction log from users so that it can handle cold start user. On the other hand, our work does not depend on mapping item mentions in user utterances to unique item identifiers which means we are able to extract user preferences directly from natural language conversation. The key idea of PMT is to leverage a key-value memory structure to transfer prior knowledge of natural language and item representations/preferences to the conversational domain. We also implement the response generation module of the Conversational Recommendation Systems using retrieval based models. The experimental results on two public conversational testbeds reveal the advantages of PMT to

learn an effective user representation in an online setting as the conversation progresses leading to better recommendations.

6.2 Future Work

6.2.1 Joint Optimization of Different Modules

In most contexts, the different modules of CRS, including the language understanding and generation task, recommendation module and user modeling system in CRSs are usually studied separately. The three components share certain objectives and data with each other [12, 55, 44, 105]. They have the exclusive data that does not benefit each other. For instance, the user interface may use the rich semantic information in reviews but not shares with a recommendation engine [47]. Besides, the two components may work in the end-to-end framework that lacks an explicit conversation strategy to coordinate them in the multi-turn conversation [47, 12], and thus the performance is not satisfied in human evaluation [35].

6.2.2 Explainability

Very few works are based on concepts and insights from Conversation Analysis, Communication Theory or related fields [86]. In some works, recommendation dialogues were discussed at a qualitative level. What seems to be mostly missing so far, however,

is a clearer understanding of what makes a CRS truly helpful, what users expect from such a system, what makes them fail [62], and which intents we should or must support in a system. Explanations are often considered as a main feature for a convincing dialogue, but these aspects are not explored a lot. In addition, more research is required to understand the mechanisms that increase the adoption of CRS, e.g., by increasing the user’s trust and developing intimacy, or by adapting the communication style (e.g., with respect to the initiative and language) to the individual user.

6.2.3 Sophisticated Strategies for Multi-turn Conversation

The multi-turn conversation strategies used in current CRS studies have more scope to implement more complex methodologies. The studies based on end-to-end dialogue systems do not even have an explicit strategy to control the multi-turn conversation [47, 12]. For example rejection of a item by user may be due to reasons other than disliking the item, like they might have already known the item. To overcome this problem, the model should consider more sophisticated strategies such as recognizing reliable negative samples. Some of the works utilized Reinforcement-Learning as a multi-turn conversation strategy by determining model actions whether to ask of recommend [85]. Meta-RL [23] can be adopted in CRSs, where the interaction is sparse and various, to speed up the training process and to improve the learning efficiency for novel subsequent tasks.

6.2.4 Knowledge Enrichment

Incorporating new/additional knowledge base will naturally improve the performance of any data-dependent system. In early stages of the development of CRSs, only the recommended items themselves were considered [14]. Later, the attribute information of items was introduced to assist in modeling user preferences [15]. Even more recent studies consider the rich semantic information in knowledge graphs [45, 105]. For example, to better understand concepts in a sentence such as “I am looking for scary movies similar to Paranormal Activity (2007)”, Zhou et al. [105] propose to incorporate two external knowledge graphs (KGs): one word-oriented KG providing relations (e.g., synonyms, antonyms, or co-occurrence) between words so as to comprehend the concept “scary” in the sentence; one item-oriented KG carrying structured facts regarding the attributes of items. In addition to the knowledge graphs, multimodal data can also be boost the performance of text-based CRS. There are some studies that exploit the visual modality, i.e., images, in dialogue systems. For example, Tong et al. [99] propose a visual dialog augmented CRS model. The model will recommend a list of items in photos, and the user will give text-based comments as feedback. The image not only helps the model learn a more informative representation of entities, but also enable the system to better convey information to the user.

Bibliography

- [1] Zahra Ashktorab, Mohit Jain, Q Vera Liao, and Justin D Weisz. Resilient chatbots: Repair strategy preferences for conversational breakdowns. In *Proceedings of the 2019 CHI conference on human factors in computing systems*, pages 1–12, 2019.
- [2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.
- [3] Nicholas J Belkin, Colleen Cool, Adelheit Stein, and Ulrich Thiel. Cases, scripts, and information-seeking strategies: On the design of interactive information retrieval systems. *Expert systems with applications*, 9(3):379–395, 1995.
- [4] Keping Bi, Qingyao Ai, Yongfeng Zhang, and W Bruce Croft. Conversational product search based on negative feedback. In *Proceedings of the 28th acm international conference on information and knowledge management*, pages 359–368, 2019.

-
- [5] Antoine Bordes, Y-Lan Boureau, and Jason Weston. Learning end-to-end goal-oriented dialog. In *ICLR*, 2017.
 - [6] Konstantinos Bousmalis, George Trigeorgis, Nathan Silberman, Dilip Krishnan, and Dumitru Erhan. Domain separation networks. In *Advances in neural information processing systems*, pages 343–351, 2016.
 - [7] John S Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. *arXiv preprint arXiv:1301.7363*, 2013.
 - [8] Giuseppe Carenini, Jocelyin Smith, and David Poole. Towards more conversational and collaborative recommender systems. In *Proceedings of the 8th international conference on Intelligent user interfaces*, pages 12–18, 2003.
 - [9] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. Universal sentence encoder. *arXiv preprint arXiv:1803.11175*, 2018.
 - [10] Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*, 2017.
 - [11] Sarath Chandar, Sungjin Ahn, Hugo Larochelle, Pascal Vincent, Gerald Tesauero, and Yoshua Bengio. Hierarchical memory networks. *arXiv preprint arXiv:1605.07427*, 2016.

-
- [12] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. Towards knowledge-based recommender dialog system. *arXiv preprint arXiv:1908.05391*, 2019.
- [13] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.
- [14] 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, 2016.
- [15] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed Huai hsin Chi. Q&r: A two-stage approach toward interactive recommendation. In *KDD*, 2018.
- [16] W Bruce Croft and Roger H Thompson. I3r: A new approach to the design of document retrieval systems. *Journal of the american society for information science*, 38(6):389–404, 1987.
- [17] Chen Cui, Wenjie Wang, Xuemeng Song, Minlie Huang, Xin-Shun Xu, and Liqiang Nie. User attention-guided multimodal dialog systems. In *SIGIR*, 2019.

-
- [18] Maurizio Ferrari Dacrema, Paolo Cremonesi, and Dietmar Jannach. Are we really making much progress? a worrying analysis of recent neural recommendation approaches. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 101–109, 2019.
- [19] Mayur Datar, Nicole Immorlica, Piotr Indyk, and Vahab S Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Proceedings of the twentieth annual symposium on Computational geometry*, pages 253–262, 2004.
- [20] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
- [21] Linus W Dietz, Saadi Myftija, and Wolfgang Wörndl. Designing a conversational travel recommender system based on data-driven destination characterization. In *ACM RecSys workshop on recommenders in tourism*, pages 17–21, 2019.
- [22] 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. In *ICLR*, 2016.
- [23] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl 2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.
- [24] Travis Ebesu, Bin Shen, and Yi Fang. Collaborative memory network for recommendation systems. In *SIGIR*, 2018.

-
- [25] Michael D Ekstrand, John T Riedl, and Joseph A Konstan. *Collaborative filtering recommender systems*. Now Publishers Inc, 2011.
- [26] Cédric Févotte and Jérôme Idier. Algorithms for nonnegative matrix factorization with the β -divergence. *Neural computation*, 23(9):2421–2456, 2011.
- [27] Chen Gao, Xiangning Chen, Fuli Feng, Kai Zhao, Xiangnan He, Yong Li, and Depeng Jin. Cross-domain recommendation without sharing user-relevant data. In *The World Wide Web Conference, WWW '19*, 2019.
- [28] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. Advances and challenges in conversational recommender systems: A survey. *arXiv preprint arXiv:2101.09459*, 2021.
- [29] Fatih Gedikli, Dietmar Jannach, and Mouzhi Ge. How should I explain ? a comparison of different explanation types for recommender systems. *International Journal of Human-Computer Studies*, 72(4):367–382, 2014.
- [30] M Goker and Cynthia Thompson. The adaptive place advisor: A conversational recommendation system. In *Proceedings of the 8th German Workshop on Case Based Reasoning*, pages 187–198. Citeseer, 2000.
- [31] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *TiiS*, 2015.

-
- [32] Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gaspiretti, Alessandro Micarelli, and Joeran Beel. Bert, elmo, use and infersent sentence encoders: The panacea for research-paper recommendation? In *RecSys*, pages 6–10, 2019.
 - [33] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *WWW*, 2017.
 - [34] Guangneng Hu, Yu Zhang, and Qiang Yang. Mtnet: A neural approach for cross-domain recommendation with unstructured text. In *KDD*, 2018.
 - [35] Dietmar Jannach and Ahtsham Manzoor. End-to-end learning for conversational recommendation: A long way to go? In *IntRS@ RecSys*, pages 72–76, 2020.
 - [36] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. A survey on conversational recommender systems. *arXiv preprint arXiv:2004.00646*, 2020.
 - [37] Heishiro Kanagawa, Hayato Kobayashi, Nobuyuki Shimizu, Yukihiro Tagami, and Taiji Suzuki. Cross-domain recommendation via deep domain adaptation. In *ECIR*. Springer, 2019.
 - [38] Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul Crook, Y-Lan Boureau, and Jason Weston. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. *arXiv preprint arXiv:1909.03922*, 2019.

-
- [39] Jie Kang, Kyle Condiff, Shuo Chang, Joseph A. Konstan, Loren G. Terveen, and F. Maxwell Harper. Understanding how people use natural language to ask for recommendations. In *RecSys*, 2017.
- [40] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3128–3137, 2015.
- [41] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- [42] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.
- [43] Ankit Kumar, Ozan Irsoy, Peter Ondruska, Mohit Iyyer, James Bradbury, Ishaan Gulrajani, Victor Zhong, Romain Paulus, and Richard Socher. Ask me anything: Dynamic memory networks for natural language processing. In *International conference on machine learning*, pages 1378–1387. PMLR, 2016.
- [44] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. Estimation-action-reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 304–312, 2020.
- [45] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. Interactive path reasoning on graph for conversational

- recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2073–2083, 2020.
- [46] Pan Li and Alexander Tuzhilin. Ddtcdr: Deep dual transfer cross domain recommendation. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 331–339, 2020.
- [47] Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. Towards deep conversational recommendations. In *Advances in Neural Information Processing Systems 31 (NIPS 2018)*, 2018.
- [48] Seth Siyuan Li and Elena Karahanna. Online recommendation systems in a b2c e-commerce context: a review and future directions. *Journal of the Association for Information Systems*, 16(2):2, 2015.
- [49] Wootack Lim, Daeyoung Jang, and Taejin Lee. Speech emotion recognition using convolutional and recurrent neural networks. In *2016 Asia-Pacific signal and information processing association annual summit and conference (APSIPA)*, pages 1–4. IEEE, 2016.
- [50] Meng Liu, Jianjun Li, Guohui Li, and Peng Pan. Cross domain recommendation via bi-directional transfer graph collaborative filtering networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 885–894, 2020.

-
- [51] Zeming Liu, Haifeng Wang, Zheng-Yu Niu, Hua Wu, Wanxiang Che, and Ting Liu. Towards conversational recommendation over multi-type dialogs. *arXiv preprint arXiv:2005.03954*, 2020.
- [52] Benedikt Loepp, Tim Hussein, and Jüergen Ziegler. Choice-based preference elicitation for collaborative filtering recommender systems. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 3085–3094, 2014.
- [53] Stanley Loh, Daniel Lichtnow, Adriana Justin Cerveira Kampff, and José Palazzo Moreira de Oliveira. Recommendation of complementary material during chat discussions. *Knowledge Management & E-Learning: An International Journal*, 2(4):385–399, 2010.
- [54] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. Content-based recommender systems: State of the art and trends. *Recommender systems handbook*, pages 73–105, 2011.
- [55] Wenchang Ma, Ryuichi Takanobu, Minghao Tu, and Minlie Huang. Bridging the gap between conversational reasoning and interactive recommendation. *arXiv preprint arXiv:2010.10333*, 2020.
- [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, 2009.

-
- [57] Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. *Introduction to information retrieval*. Cambridge university press Cambridge, 2008.
- [58] Ahtsham Manzoor and Dietmar Jannach. Generation-based vs. retrieval-based conversational recommendation: A user-centric comparison. In *Fifteenth ACM Conference on Recommender Systems*, pages 515–520, 2021.
- [59] Ahtsham Manzoor and Dietmar Jannach. Towards retrieval-based conversational recommendation. *arXiv preprint arXiv:2109.02311*, 2021.
- [60] Lorraine McGinty and Barry Smyth. On the role of diversity in conversational recommender systems. In *International Conference on Case-Based Reasoning*, pages 276–290. Springer, 2003.
- [61] Alexander H. Miller, Adam Fisch, Jesse Dodge, Alireza Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. In *EMNLP*, 2016.
- [62] Mohammed Slim Ben Mimoun, Ingrid Poncin, and Marion Garnier. Case study—embodied virtual agents: An analysis on reasons for failure. *Journal of Retailing and Consumer services*, 19(6):605–612, 2012.
- [63] Kaixiang Mo, Yu Zhang, Shuangyin Li, Jiajun Li, and Qiang Yang. Personalizing a dialogue system with transfer reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

-
- [64] Ali Montazeralghaem, James Allan, and Philip S Thomas. Large-scale interactive conversational recommendation system using actor-critic framework. In *Fifteenth ACM Conference on Recommender Systems*, pages 220–229, 2021.
- [65] David Nadeau and Satoshi Sekine. A survey of named entity recognition and classification. *Linguisticae Investigationes*, 30(1):3–26, 2007.
- [66] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.
- [67] Fedelucio Narducci, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. Improving the user experience with a conversational recommender system. In *International Conference of the Italian Association for Artificial Intelligence*, pages 528–538. Springer, 2018.
- [68] Thuy Ngoc Nguyen and Francesco Ricci. A chat-based group recommender system for tourism. *Information Technology & Tourism*, 18(1):5–28, 2018.
- [69] Liqiang Nie, Wenjie Wang, Richang Hong, Meng Wang, and Qi Tian. Multimodal dialog system: Generating responses via adaptive decoders. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 1098–1106, 2019.
- [70] Gustavo Penha and Claudia Hauff. What does bert know about books, movies and music? probing bert for conversational recommendation. In *Fourteenth ACM Conference on Recommender Systems*, pages 388–397, 2020.

- [71] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. Deep contextualized word representations. In *NAACL-HLT*, 2018.
- [72] Minghui Qiu, Feng-Lin Li, Siyu Wang, Xing Gao, Yan Chen, Weipeng Zhao, Haiqing Chen, Jun Huang, and Wei Chu. Alime chat: A sequence to sequence and rerank based chatbot engine. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 498–503, 2017.
- [73] Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K Lam, Sean M Mc-Nee, Joseph A Konstan, and John Riedl. Getting to know you: learning new user preferences in recommender systems. In *Proceedings of the 7th international conference on Intelligent user interfaces*, pages 127–134, 2002.
- [74] Xuhui Ren, Hongzhi Yin, Tong Chen, Hao Wang, Zi Huang, and Kai Zheng. Learning to ask appropriate questions in conversational recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 808–817, 2021.
- [75] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-thieme. BPR : Bayesian Personalized Ranking from Implicit Feedback. *UAI*, 2009.
- [76] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*, 2012.

-
- [77] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.
- [78] Francesco Ricci, Lior Rokach, Bracha Shapira, and Paul B. Kantor. Recommender systems handbook. 2015.
- [79] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *Advances in Neural Information Processing Systems*, volume 20, 2008.
- [80] Yasser Salem, Jun Hong, and Weiru Liu. History-guided conversational recommendation. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 999–1004, 2014.
- [81] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [82] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295, 2001.
- [83] Hideo Shimazu. Expertclerk: a conversational case-based reasoning tool for developing salesclerk agents in e-commerce webshops. *Artificial Intelligence Review*, 18(3):223–244, 2002.

-
- [84] Alessandro Suglia, Claudio Greco, Pierpaolo Basile, Giovanni Semeraro, and Annalina Caputo. An automatic procedure for generating datasets for conversational recommender systems. In *CLEF*, 2017.
- [85] Yueming Sun and Yi Zhang. Conversational recommender system. In *SIGIR*, pages 235–244. ACM, 2018.
- [86] Paul Thomas, Mary Czerwinski, Daniel McDuff, and Nick Craswell. Theories of conversation for conversational ir. *ACM Transactions on Information Systems (TOIS)*, 39(4):1–23, 2021.
- [87] Cynthia A Thompson, Mehmet H Goker, and Pat Langley. A personalized system for conversational recommendations. *Journal of Artificial Intelligence Research*, 21:393–428, 2004.
- [88] Daisuke Tsumita and Tomohiro Takagi. Dialogue based recommender system that flexibly mixes utterances and recommendations. In *2019 IEEE/WIC/ACM International Conference on Web Intelligence (WI)*, pages 51–58. IEEE, 2019.
- [89] Robin Van Meteren and Maarten Van Someren. Using content-based filtering for recommendation. In *Proceedings of the machine learning in the new information age: MLnet/ECML2000 workshop*, volume 30, pages 47–56, 2000.
- [90] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

-
- [91] Maksims Volkovs, Guang Wei Yu, and Tomi Poutanen. Dropoutnet: Addressing cold start in recommender systems. In *NIPS*, 2017.
 - [92] Huazheng Wang, Qingyun Wu, and Hongning Wang. Factorization bandits for interactive recommendation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.
 - [93] Jiaqi Wang and Jing Lv. Tag-informed collaborative topic modeling for cross domain recommendations. *Knowledge-Based Systems*, 203:106119, 2020.
 - [94] Pontus Wärnestål. User evaluation of a conversational recommender system. In *Proceedings of the 4th Workshop on Knowledge and Reasoning in Practical Dialogue Systems*, 2005.
 - [95] 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. In *ACL*, 2017.
 - [96] Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart Van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.
 - [97] Qingyun Wu, Naveen Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 495–504, 2018.

-
- [98] 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.
- [99] Tong Yu, Yilin Shen, and Hongxia Jin. A visual dialog augmented interactive recommender system. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 157–165, 2019.
- [100] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.
- [101] Jie Zeng, Yukiko I Nakano, Takeshi Morita, Ichiro Kobayashi, and Takahira Yamaguchi. Eliciting user food preferences in terms of taste and texture in spoken dialogue systems. In *Proceedings of the 3rd International Workshop on Multisensory Approaches to Human-Food Interaction*, pages 1–5, 2018.
- [102] 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):1–38, 2019.
- [103] Yinan Zhang, Yong Liu, Peng Han, Chunyan Miao, Lizhen Cui, Baoli Li, and Haihong Tang. Learning personalized itemset mapping for cross-domain recommendation. In *IJCAI*, pages 2561–2567, 2020.

-
- [104] 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, 2018.
- [105] Kun Zhou, Wayne Xin Zhao, Shuqing Bian, Yuanhang Zhou, Ji-Rong Wen, and Jingsong Yu. Improving conversational recommender systems via knowledge graph based semantic fusion. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1006–1014, 2020.
- [106] Feng Zhu, Yan Wang, Chaochao Chen, Guanfeng Liu, and Xiaolin Zheng. A graphical and attentional framework for dual-target cross-domain recommendation. In *IJCAI*, pages 3001–3008, 2020.
- [107] Yu Zhu, Yu Gong, Qingwen Liu, Yingcai Ma, Wenwu Ou, Junxiong Zhu, Beidou Wang, Ziyu Guan, and Deng Cai. Query-based interactive recommendation by meta-path and adapted attention-gru. In *CIKM*, 2019.
- [108] Jie Zou, Yifan Chen, and Evangelos Kanoulas. Towards question-based recommender systems. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 881–890, 2020.
- [109] Lixin Zou, Long Xia, Yulong Gu, Xiangyu Zhao, Weidong Liu, Jimmy Xiangji Huang, and Dawei Yin. Neural interactive collaborative filtering. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 749–758, 2020.