# Natural Language Explanation for Recommendations and Beyond

**LI Lei**

**A thesis submitted in partial fulfilment of the requirements**
**for the degree of**
**Doctor of Philosophy**

**Principal Supervisor:**
**Dr. CHEN Li (Hong Kong Baptist University)**

**May 2022**

# DECLARATION

I hereby declare that this thesis represents my own work which has been done after registration for the degree of PhD at Hong Kong Baptist University, and has not been previously included in a thesis or dissertation submitted to this or any other institution for a degree, diploma or other qualifications.

I have read the University's current research ethics guidelines, and accept responsibility for the conduct of the procedures in accordance with the University's Research Ethics Committee (REC). I have attempted to identify all the risks related to this research that may arise in conducting this research, obtained the relevant ethical and/or safety approval (where applicable), and acknowledged my obligations and the rights of the participants.

Signature: Lei Li

May 2022

# Abstract

Explainable artificial intelligence (XAI) could unveil the decisions made by AI, help human understand machine behavior, and further build trust between end users and AI systems. However, most existing XAI approaches are primarily mathematical, and also specialized to domain experts. Although they can help researchers understand the underlying working mechanism of AI models, the explanations are hardly comprehensible to ordinary users. As a user-centric application, recommender systems interact with and serve a great number of such users. As a consequence, how to explain recommended products to them, in order to help them make informed choices and provide better service, becomes a critical and practical problem.

As a primary media of communication, language can be easily understood by almost everyone. Therefore, natural language explanation for recommendations has gained increasing attention recently. Despite of that, little work has been done to provide explanations from the perspective of a user's changing context, such as companion and destination if the recommendation is a hotel. To fill this research gap, we have devised a new context-aware recommendation approach that particularly matches latent features to explicit contextual features for producing context-aware explanations. But the explanation format in this approach, as with many previous works, is limited to predefined templates, which could restrict explanation expressiveness, and thus may not be able to well explain the specialty of a recommendation. In an attempt to further enrich explanation expressiveness and quality, we have proposed a neural template generation approach that can learn templates from data. In order to generate such template-like explanations, an item feature must be specified

in advance, either automatically or manually. To accommodate situations where features are unavailable, we have designed a more general method that can generate natural language explanations with or without features.

The proceeding two methods can generate high-quality explanations, but do not always guarantee factual correctness, e.g., "four-horned unicorns" as produced by a well-known pre-trained language model. To cope with this problem, we wonder whether explanations for a recommendation could be ranked, as if they are web pages returned by a search engine in accordance with a given query, saving the need to worry about the content of explanations. To enable such explanation ranking, we have created benchmark datasets by automatically identifying nearly identical sentences across different user reviews, based on the wisdom of the crowd. In addition, the ranking formulation makes it possible for standard evaluation of explanations via ranking-oriented metrics. Based on this, we have studied if purposely selecting some explanations could reach certain goals, e.g., improving recommendation accuracy. This could potentially lead to unfaithful explanations that attempt to lure users' clicking and purchasing. Hence, at the end of this thesis, we discuss this type of unintentionally negative effects as well as other open issues in explainable recommendations, such as bias and fairness.

We believe that these works conducted in this thesis are non-trivial and meaningful to the community of XAI. They are instantiated on the scenario of explainable recommendations, but could be generalized to a broader scope of fields in AI, e.g., dialogue systems.

**Keywords:** Explainable Recommendation, Explainable Artificial Intelligence, Recommender Systems, Context-aware Recommender Systems, Natural Language Generation, Learning to Rank

# Acknowledgments

I would like to pay tribute to the following people who had an important part in my Ph.D. journey.

First and foremost, I want to express my sincere gratitude to my supervisor Dr. Li Chen for making the past five years of doing research fun and enjoyable, though a bit challenging. Before my Ph.D., I literally had no research experience, so she had to teach me everything from the very beginning with great patience. She gave me the freedom to explore what I am truly passionate about, and meanwhile reminded me promptly when I was struggling with research problems in a wrong way. She values my thoughts and opinions, while she may hold a different view. She polished the papers which I wrote so carefully and artfully that after attempting to learn from her sometimes when I write I feel like I am using her hand. All these that she left on me would benefit the rest of my life.

Again, thanks to Dr. Chen, I am very lucky to get to know Dr. Yongfeng Zhang and Dr. Ruihai Dong, two leading scholars in our field. Dr. Zhang is extremely knowledgeable and visionary. After years of having regular online meetings with him, my horizon is widely broadened and my research taste is well cultivated. Whenever I propose something, he puts it in a bigger picture and gives me so many insightful comments. Moreover, we both have the ambition and intention to do meaningful and impactful things that have long-lasting value. I am deeply grateful to him for the guidance and support. Likewise, I am quite fortunate to meet Dr. Dong, who guided me through trending technologies at the early stage of my Ph.D., and advised me to embrace them. He also taught me how to write and present a paper,

and how to respond to reviews. In addition to research, I am thankful to him for the encouragement, when I was under a lot of pressure and started to question the meaning of everything.

It is my great honor to have Prof. Guibing Guo and Prof. Piji Li as my thesis examiners. Both of them gave me valuable feedback to further improve my thesis. Besides, part of it is actually built on top of Prof. Li's innovative research. I also thank Prof. Tiejun Tong for chairing my oral examination, as well as my thesis committee members Prof. Yiu-ming Cheung and Dr. Jing Ma for their time and efforts. In particular, this thesis is quite relevant to Dr. Ma's research, for which I once had a really enlightening discussion with her about natural language processing.

In our Department of Computer Science, I would like to thank the professors and staff for creating a wonderful environment, especially Prof. William Cheung who gave me a lot of helpful advice on teaching and kindly allowed me to join his research group meeting, and Kristina Tsang who so attentively dealt with all the matters related to my research.

When I was an undergraduate student, I majored in two bachelor degrees, and therefore had two supervisors. Both of them had a huge impact on me, so I would like to thank them here. My genuine thanks go to Dr. Weike Pan, who saw the potential in me and offered me a life-changing opportunity. Without his referral to Dr. Chen, I would never have any idea of doing a Ph.D. let alone finish this thesis. He also systematically taught me all the fundamental algorithms in recommender systems and information retrieval, which largely facilitated my research progress. In the meantime, I thank Dr. Guo Li for the enlightenment, as well as the grand goal that he set up for me to achieve. Each year before COVID-19, I pay him a visit, and we usually talk for hours. As a mathematician, he often shares with me his insights about research, such as "what in prior work are not always correct" and "publication is one thing; pioneering work is quite another", which I have been turning over in my mind.

Owing to the English Language Enhancement Service provided by Language

Center, I met two nice and amazing English teachers. Thanks to Dr. Cissy Li for teaching me a number of writing techniques, such as readership awareness and analysis depth, which greatly improved my academic writing. She also patiently corrected many of my writing errors that I had been fully unaware of. At the same time, Dr. Meilin Chen helped with my colloquial English. I was not very confident and was too afraid of making mistakes. She helped me conquer the fear of speaking in public. In addition, I would like to thank her for encouraging me to do something that I hesitated about, and for sharing life experience with me when I was in a difficult time.

Throughout the years in HKBU, many labmates and colleagues offered me great help. Thanks to Dr. Wen Wu for helping me in almost every way when I first came to Hong Kong, a totally new environment. Thanks to Dr. Feng Wang for nicely providing technical support, when I was overwhelmed with errors and bugs. Thanks to Dr. Lin Zheng and Dr. Hechang Chen for giving me useful suggestions, especially those regarding sleeping and health which I did not pay much attention to when I was younger, and did result in some problems. Thanks to Dr. Qi Tan, Dr. Qing Cai, Dr. Huiqi Deng, Chuyu Liu, Dr. Guozhong Li, Wanling Cai, Ningxia Wang, Zhirun Zhang, Yongxin Ni, Riwei Lai, Xianglin Zhao and Dr. Yucheng Jin for the great time that we spent together. Special thanks to my roommate Jinfu Ren for everything that he has done for me.

During my Ph.D., I did an unforgettable internship at Inspir.ai, where my mind was really opened up, especially when I was asked to read research papers that advanced cutting-edge technology, and when I witnessed how a group of brilliant minds built AI systems to beat professional computer game players. Thank you to Dr. Peng Peng and Quan Yuan for their mentorship, and to Rong Hu and Juanjuan Hu for their tremendous help.

Between research and life, I occasionally find a balance, ever since I met the following extraordinary mates who know exceedingly well how to enjoy life and have fun. Zexiong Cai completely opened up a new world for me, when he took me to

various places to appreciate the beauty of nature. Later, with brave Dr. Yixin Zhou I climbed a huge amount of challenging mountains and exciting cliffs, from which I found supreme happiness. Chi Hiu Mung and I practiced archery together for half a year. Dr. Chao Yu helped me improve my swimming skill hand by hand. There are many playmates whom I usually do exercises or outdoor activities with: Feng Yu, Chunpeng Zhou, Dr. Jiaxu Cui, Chenyin Liu, Longxu Sun, Jingtao Zou and Fanhui Pei. I thank them all for the joy and delights that we had together.

I appreciate and cherish the friendship with the following peers. I am indebted to Miao Liu who always reaches out whenever I need help, and always shares with me his latest findings from both academia and industry. It feels comfortable to have someone to chat about anything no matter how far away we are from each other, for that I would like to thank Fen Cao, Hongkuan Zhang, Wei Dai, Lin Li and Dugang Liu. A big thank you to Zixiang Chen, Zijie Zeng, Yaofeng Chen, Qing Zhang and Jinchun Wen for the generous help that they offered to me when I was in SZU.

Lastly, I would like to thank my parents and grandparents for raising me, and for always giving me encouragement as well as support to my decisions since I was little.

# Table of Contents

# List of Tables

# List of Figures

# List of Algorithms

# Chapter 1

# Introduction

## 1.1 Motivation

With the ever growing computing capability of computers as well as the unprecedented advancement of deep neural networks, artificial intelligence (AI) models are getting more and more capable of making predictions, such as AlexNet [58] for visual recognition and BERT [30] for natural language understanding. However, it is difficult even for their designers to answer why a particular prediction in these models is made, owing to the sophisticated model structure as compared with traditional linear models, e.g., linear regression. In fact, unveiling the decisions could help people better understand model behavior, and further enable trust between human and machines, especially in high-stake scenario, e.g., health care. Hence, there emerges a new research direction: explainable AI (XAI) [42], which attempts to make AI systems interpretable to human being. Although existing XAI approaches, such as LIME [101] and SHAP [82], could help researchers comprehend AI models' internal working mechanism, they are very mathematical and somewhat limited to domain experts. This thesis attempts to address this problem by producing natural language explanations that are understandable to ordinary people, but its key focus is on recommender systems, a sub-field of AI.

Recommender systems have been widely deployed on online commercial plat-

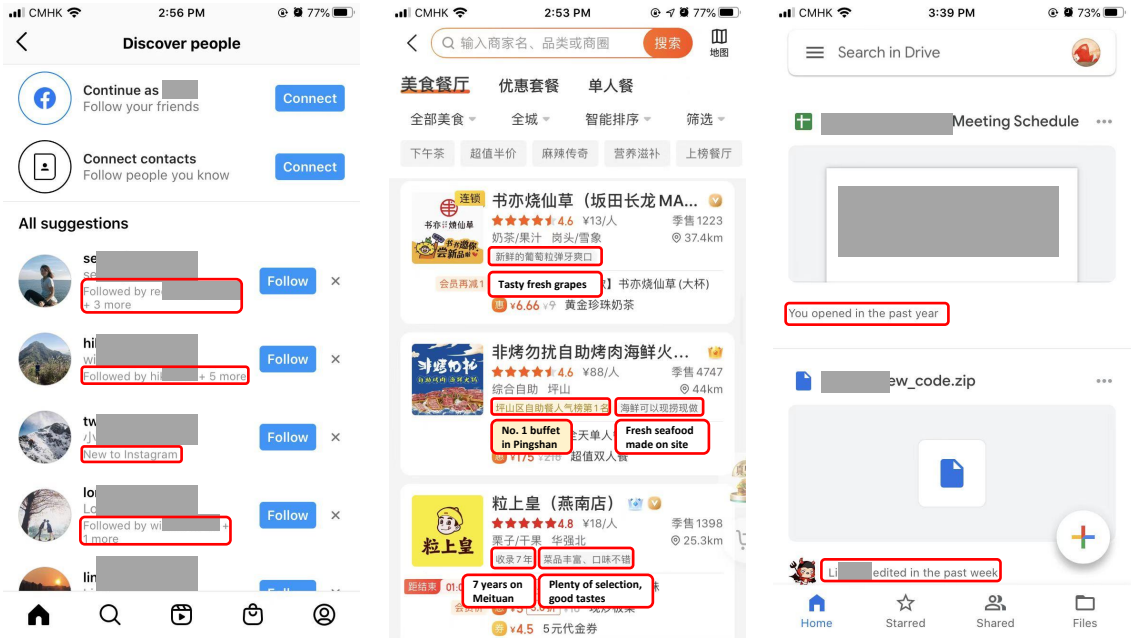



Figure 1.1: Three examples of recommendation explanation on commercial apps, for respectively social recommendation (Instagram), food recommendation (Meituan) and document recommendation (Google Drive). For privacy protection, some areas are shaded.

forms, ranging from e-commerce, video-sharing to social media, e.g., Amazon[1], Youtube[2] and Instagram[3], because they can tackle the information overload problem by returning personalized products or information to users. Meanwhile, they can also benefit service providers, e.g., increasing their revenue. Over the years, there emerges a variety of recommendation algorithms, including collaborative filtering (CF) [103, 100], matrix factorization (MF) [88, 57] and deep neural networks [45, 129]. Though most of them have been demonstrated effective, it is difficult for the latter two types of method to illustrate why a product is recommended, since they represent users and items as latent factors/vectors that cannot directly reflect users' preferences towards explicit item features.

Recently, explainable recommendation has drawn increasing attention from both academia [111, 131] and industry [126, 127], because it has been shown that pro-

[1]https://www.amazon.com
[2]https://www.youtube.com
[3]https://www.instagram.com

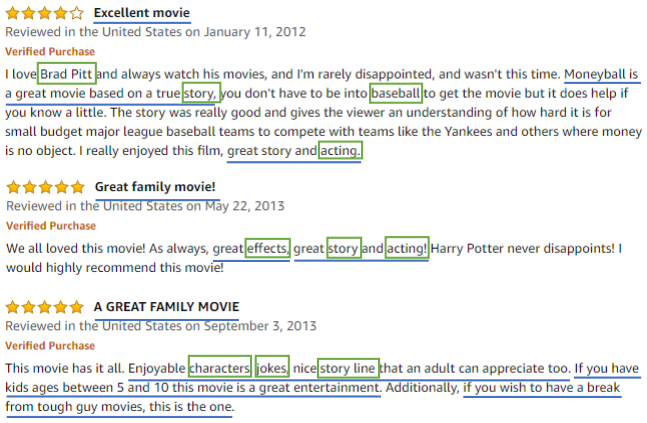



Figure 1.2: Three user reviews for different movies from Amazon (Movies & TV category). Sentences of explanation purpose are underlined, while features that can be used for explanation are in rectangles.

viding appropriate explanations can help users make better and/or faster decisions, increase the system's ease of use and enjoyment, and gain user trust in the system [111, 131]. In fact, explanation is as important as recommendation itself, because there is usually no absolute "right" or "wrong" in terms of which item(s) to recommend. Instead, multiple items may all be of interest to a user, and it all depends on how they are explained to the user. Among various explanation styles (e.g., images [22, 38] and item neighbors [71, 93]), natural language explanation is more preferred by commercial platforms (see Fig. 1.1), owing to its advantage in communicating rich information. However, there are three important issues yet to be addressed in explainable recommendation research: **benchmark datasets**, **effective approaches** and **quantitative evaluation**.

In terms of data, user reviews have been widely utilized for providing natural language explanations [133, 43, 13, 81, 14, 36, 116, 119, 24, 16, 26], because they contain users' real feedback towards items, and thus are an ideal proxy of explanation. However, simply adopting reviews [14, 34, 81, 109] or their first sentences [26] as explanations is less appropriate, because some sentences are less suitable for explanation purpose (see the first review's first sentence in Fig. 1.2). We argue that a more appropriate way is to identify some item features (e.g., "acting") or sentences (e.g., "great story and acting") that can serve as explanations (see the highlights in

Fig. 1.2). With this in mind, we create two types of benchmark datasets respectively for natural language explanation generation and explanation ranking.

In terms of methodology, there are several critical problems that are mostly ignored by existing explainable recommendation approaches. We briefly summarize them below, and discuss and address them with more details in the corresponding chapters.

- Explanations produced by existing approaches are mostly static, and therefore hardly adaptable to a user's changing contexts, such as location and time, which are quite important particularly when a user is actively looking for service-oriented recommendation, e.g., restaurant.

- The template based explanation style as adopted in most existing approaches could not always well explain a recommendation, since its "backbone" is fixed permanently and may lose flexibility, e.g., "You might be interested in [feature], on which this product performs well" [133, 36, 110].

- The semantic gap between IDs (indispensable in recommender systems) and words makes it difficult to unleash the language modeling power of the well-known Transformer model [114] for explanation generation, as well as other relevant natural language generation tasks, such as review summarization and dialogue systems.

- The potential impact of explanations to recommendations is rarely studied in existing approaches, because their explanations are often side outputs of recommendation models. This could also be attributed to the lack of standard evaluation of explainability, since the explanations vary from model to model, i.e., model-dependent.

Evaluation of explanations in existing works can be generally classified into four categories, including case study, user study, online evaluation and offline evaluation [131]. In most works, case study is adopted to show how the example explanations are correlated with recommendations. These examples may look intuitive, but they

are less representative to reflect the overall quality of the explanations. Results of user study [5, 39] are more plausible, but it can be expensive and is usually evaluated in simulated environments which may not always reflect real users' actual perception. Although this is not a problem in online evaluation, it is difficult to implement as it relies on the collaboration with industrial firms, which may explain why only few works [133, 126, 84] conducted online evaluation. Consequently, one may wonder whether it is possible to evaluate the explainability using offline metrics. However, as far as we know, there is no standard metrics that are well recognized by the community. Although BLEU [91] in machine translation and ROUGE [75] in text summarization have been widely adopted to evaluate text quality for natural language generation, text quality is not equal to explainability [16]. To cope with the problem, we come up with new explainability metrics that are complementary to text quality with a particular focus on item features. In addition, we propose a learning to rank formulation [79] for recommendation explanations, so as to achieve standard offline evaluation.

## 1.2 Outline

This thesis introduces three different ways to implement natural language explanation for recommendations: template-based context-aware explanation, generated textual explanation, and ranked textual explanation. The technical details are presented in four separate chapters.

In Chapter 2, we first go through an exhaustive amount of related works on explainable recommendation, context-aware recommender systems, natural language generation, and learning to rank. In the meantime, we discuss their shortcomings and how this thesis is motivated by them.

In Chapter 3, we present our explanation approach for context-aware recommendations, which we dub CAESAR, standing for Context-Aware Explanation based on Supervised Attention for Recommendations. To realize such explanation, we also come up with a feature mining approach to discover high-quality contextual features

from user reviews. This chapter is based on the work published at JIIS [61].

In Chapter 4, we develop a NEural TEmplate (NETE) explanation generation approach that is able generate template-shaped explanations. Besides the model, we provide public datasets for natural language explanation generation, and offer the way to create them. In addition, we propose four novel metrics to evaluate the explainability of generated explanations from the perspective of item features and sentence diversity. This chapter is based on the work published at WWW'20 (demo) [62] and CIKM'20 [64].

In Chapter 5, we describe a PErsonalized Transformer for Explainable Recommendation (PETER), where we design an elegant task to address the problem that vanilla Transformer, though being demonstrated with strong language modeling capability, fails to make use of user and item IDs (which are essential in recommender systems) for natural language generation. This chapter is based on the work published at ACL'21 [67].

In Chapter 6, we argue that explanations could be ranked just like documents returned by search engines, which would enable standard evaluation of explainability. To this end, we present three benchmark datasets for EXplanaTion RAnking (EXTRA) created by conducting near-duplicate detection on user reviews. We address the inherent sparsity issue in the user-item-explanation interaction data, by treating the ternary relationship as two groups of binary relationships. Further, we extend traditional item ranking to an item-explanation joint-ranking formalization to show that purposely selecting some particular explanations could lead to improved recommendation accuracy. Part of the work in this chapter was published at SIGIR'21 (resource) [65], and the remaining part [66] is under review at the time of thesis submission.

In Chapter 7, we conclude this thesis, and discuss open problems in explainable recommendation research, such as bias and fairness.

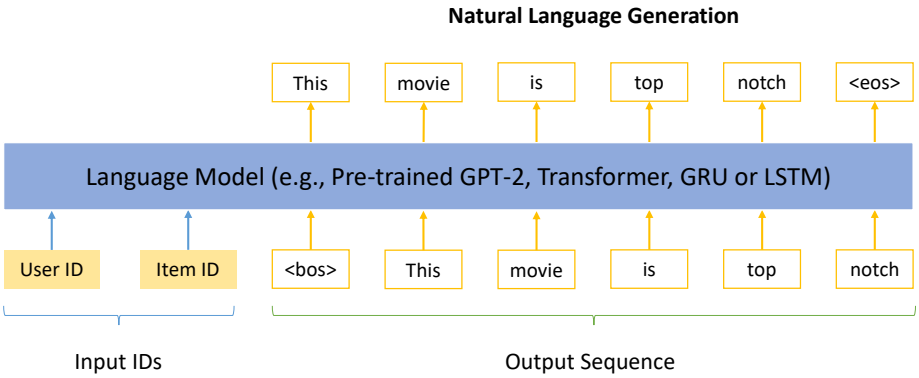



Figure 1.3: Overview of recommender systems-based natural language generation. In the case of recommendation explanation generation, the model is instructed to generate a word sequence for explaining why an item is recommended to the user.

## 1.3 Contributions

The main contributions of this thesis are listed below:

- We design a context-aware recommendation approach that is able to produce context-aware explanation, which is important particularly in mobile environment, but mostly ignored by existing works. As shown in our human evaluation, the context-aware explanations are perceived more helpful to users than context-unaware explanations.

- We present a neural template explanation generation approach, which bridges the merits of both template-based and natural language generation-based approaches for both explanation expressiveness and quality. We release our datasets, as well as the way to construct them so that other researchers can create their own datasets. For evaluation, we design four explainability metrics that emphasize more item features in generated explanations, as a complement to existing text quality measures.

- We propose a personalized Transformer that can make recommendation and generate natural language explanation simultaneously by bridging the gap between IDs (important identifiers in recommender systems) and words. On top

of the proceeding two works, we create a small ecosystem[4] to facilitate the development of recommender systems-based natural language generation (see Fig. 1.3). It consists of implementation of typical language models (including pre-trained model GPT-2 [95, 68], Transformer [114, 67], GRU [27, 70] and LSTM [47, 31]), evaluation metrics and public datasets.

- We formulate the explanation problem as a ranking-oriented task, and construct three benchmark datasets which consist of user-item-explanation interactions, with an attempt to achieve standard evaluation of explainability for explainable recommendation via well-known ranking metrics, such as NDCG, precision and recall. With the evaluation and data, we propose an item-explanation joint-ranking framework that can reach our designed goal, i.e., improving the performance of both recommendation and explanation, as evidenced by our experimental results.

[4]Available at https://github.com/lileipisces/NLG4RS

# Chapter 2

# Literature Survey

There are four lines of research closely related to this thesis: explainable recommendation, context-aware recommendation, natural language generation and learning to rank. In this chapter, we present a literature survey regarding the four branches of research work.

## 2.1 Explainable Recommendation

There are two major directions on explainable recommendation research [111, 131]: human-computer interaction and machine learning. The former [37, 19, 59, 20] investigates how people perceive different styles of explanation, while the latter provides explanations by designing new explainable recommendation algorithms, to which this thesis is more related. In these works, there is a variety of explanation styles to recommendations, including visual highlights [22, 48], textual highlights [105, 80], item features [43, 117], generated images [38], knowledge graph paths [3, 122, 35, 123, 118], reasoning rules [107, 17, 139], pre-defined templates [133, 36, 116, 110], automatically generated text [90, 26, 28, 81, 16] and retrieved text [24, 119, 14, 13], etc. The works conducted in this thesis are more relevant to the last three.

To produce template-based explanations, a number of methods has been proposed to identify keywords to be filled in the template, such as matrix factorization

[133], tensor factorization [116], and attention mechanism [36]. Our context-aware explanation approach lies in the last category because it makes use of attention mechanism, but it differs from related methods in that it produces context-aware explanations. There are few related works [7, 104] that take context into consideration for explanation purpose. However, in [7] only location based explanation is provided, so the model is restricted to one type of context, while ours can accommodate various types of context into explanation. The second work [104] is limited to item-level context-aware explanation, e.g., "recommend for use as [couples]", so the explanation cannot reason in terms of what features the recommended item would be suitable for the user's current contexts.

However, it will involve costly human efforts and domain knowledge to design templates. Moreover, since they are predefined and fixed, the diversity and flexibility of explanations in template-based approaches may be hindered. This motivates the development of natural language generation methods. Despite the popularity, researchers have pointed out that they tend to generate universal and safe sentences, which carry little meaning and thus may not be very useful to users [89, 128, 135]. With an attempt to improve the expressiveness of generated text, researchers have introduced keywords [89, 128] or a group of attributes [135] as input to their models. Meanwhile, previous methods usually adopt a user review [120, 112, 81, 109], its first sentence [26], or the review title [70] as the training data, which may not always be relevant to the recommended item. As a comparison, in our neural template generation approach, the data are review sentences that contain at least one concrete item feature. This makes the generated explanations more informative and relevant to the items, and thus could help users better understand the recommendations. Besides data, previous works mostly rely on recurrent neural networks (RNN), leaving the potentially more effective Transformer under-explored, which motivates our work of personalized Transformer. More technical details are given in the following section.

Besides automatically generating text, retrieving available text is also a popular

way to explain recommendations. For example, some retrieval-based methods [14, 13] present to users a few reviews as selected from the target item's review collection. As a user review could be very long and may contain noisy information that is not really useful [46, 119], some studies have attempted to select sentences instead of the whole review as explanations [119, 24]. But explanations in these works are merely side outputs of the recommendation models. As a result, none of these works measured the explanation quality based on benchmark metrics. In comparison, our explanation ranking approach formulates the explanation task as a learning to rank [79] problem, which enables standard offline evaluation via ranking-oriented metrics.

Regarding explanation evaluation, there are some works [26, 90] that regard text similarity metrics (i.e., BLEU [91] in machine translation and ROUGE [75] in text summarization) as explainability, when generating textual explanations for recommendations. The evaluation issue is still under debate, since text similarity does not equal to explainability [16]. For example, when the ground-truth is "sushi is good", two generated explanations "ramen is good" and "sushi is delicious" gain the same score on the two metrics. However, from the perspective of explainability, the latter is obviously more related to the ground-truth, as they both refer to the same feature "sushi", but the metrics fail to reflect this issue. This is why we propose four metrics that particularly evaluate item features and sentence diversity in our neural template generation approach, and also come up with the explanation ranking formulation that may enable standard evaluation of explanations.

## 2.2 Context-aware Recommendation

Context-aware recommendation algorithms can be broadly classified into three categories according to the phase when contextual information is incorporated: contextual pre-filtering, contextual post-filtering and contextual modeling [2]. Contextual pre-filtering approaches are able to make use of traditional recommendation algorithms, e.g., matrix factorization (MF) [88], where contexts play the role of data filtering, e.g., selecting rating data for one certain context. However, this approach

suffers from severe data sparsity problem, as the data may become sparse after filtering. Existing recommendation algorithms can also be adopted to contextual post-filtering, in which a given context is used to filter out irrelevant recommendations or adjust the recommendation list. Recently, more researchers start to investigate contextual modeling techniques. We classify these works into two groups. The first group of work has been engaged in leveraging contextual features for recommendation [15, 60], for which the recommendation process normally contains several steps, including contextual feature extraction, sentiment orientation detection, review score calculation, and item score computation. However, the whole process may involve costly human labeling efforts or require domain knowledge. In comparison, our context-aware recommendation approach is a general neural network that models contextual features without human efforts. The other difference is that this model can utilize contextual features for both recommendation and explanation, while the related methods mainly leverage contextual features for recommendation only.

The second group of work does not consider contextual features, but models contexts in various ways. For example, in [51], users, items, and contexts are all represented as factors in a user-item-context tensor, which is then factorized by Tucker decomposition [53]. In [77], tensors are also applied, but are used to derive contextual operating matrices and context-specific vectors for rating prediction. In [6], matrix factorization [88] is extended to incorporate the influence of contexts as bias parameters, which could be related to items or their categories. In factorization machines (FM) [97] based methods [124, 44, 125], users, items, and contexts are equally treated as features for sparse data prediction. Specifically, [124] leverages attention mechanism to distinguish the importance of different feature interactions, [44] applies neural networks to learn non-linear and high-order feature interactions, and [125] performs 3D convolutions on an interaction cube to capture high-order interaction signals. More recent methods have attempted to characterize the relation between users/items and contexts in an attentive manner to distinguish the impact of

different contexts on users' preference [85, 63]. The major difference between [85] and [63] is that the former adopts vanilla attention mechanism, while the latter utilizes co-attention. Although these methods achieve certain accuracy improvement, they are hardly explainable because their features are mostly in latent representations (e.g., vectors or scalars), while our proposed context-aware approach can leverage explicit contextual features for explanation purpose.

## 2.3 Natural Language Generation

In the field of natural language processing, the encoder-decoder framework has been widely deployed in many applications, such as machine translation [27], conversational systems [89, 128], and text summarization [11]. For the application of recommender systems-based natural language generation, user and item IDs are usually mandatory. Previous approaches typically adopt multi-layer perceptron (MLP) as the encoder to encode the IDs into a context vector, from which RNN, such as long short-term memory (LSTM) [47] and gated recurrent units (GRU) [27], can decode a word sequence. This strategy can be found in many applications, such as review generation [31, 120, 112], tip generation [70, 69] and explanation generation [26, 16]. Moreover, since the generation process of natural language generation methods may not be easy to control [121], some works have leveraged attention mechanism [83] or copy mechanism [41] to force the models to include some specific words. However, these techniques cannot guarantee which word to be included. In comparison, our neural template generation approach is able to place the given feature in the generated explanation, which achieves certain degree of controllability.

These techniques are developed for RNN-based approaches, so they may not be applicable to the more recent Transformer [114] that relies entirely on self-attention. This well-known model is first brought to the domain of machine translation with the architecture of encoder-decoder. Later works [76, 30] show that it remains effective, even when the encoder or the decoder is removed, reducing nearly half of the parameters. Under the paradigm of pre-training plus fine-tuning, Transformer's

effectiveness has been confirmed on a wide range of tasks, including both natural language understanding and generation [94, 30, 32]. Particularly, it is able to perform novel tasks, e.g., arithmetic, after scaling up both the model and the training data [95, 10]. However, it may not be friendly to researchers who do not possess large amounts of computing resources. Instead, our personalized Transformer explores small unpretrained models, as they are computationally cheaper and more flexible when being adapted to new applications, e.g., dialogue systems. Probably because a proper solution to deal with heterogeneous inputs (i.e., IDs and words) is yet to be found, previous works with Transformer for natural language generation replace IDs with text segments, such as persona attributes [137], movie titles [138] and item features [90], which are in the same semantic space as the word sequence to be generated. In comparison, our solution for the personalized Transformer is to design an effective task that can give the IDs linguistic meanings, thus connecting IDs with words.

## 2.4 Learning to Rank

The application of learning to rank [79] can be found in many other domains. For instance, [115, 49] attempt to explain entity relationship on knowledge graphs via ranking. The major difference from our explanation ranking for recommendations is that they heavily rely on the semantic features of explanations, either constructed manually [115] or extracted automatically [49], while one of our ranking models works well when leveraging only the relation of explanations to users and items, without considering such features.

Our proposed explanation ranking models are experimented on textual datasets, but it can be applied to a broad spectrum of other explanation styles, e.g., images. Concretely, on each dataset there is a pool of candidate explanations to be selected for each user-item pair. A recent online experiment conducted on Microsoft Office 365[1] [126] shows that this type of globally shared explanations is indeed helpful

[1]https://www.office.com

to users. The main focus of this work is to study how users perceive explanations, which is different from ours that aims to design effective models to rank explanations. Despite of that, their research findings motivate us to provide better explanations that could lead to improved recommendations.

In more details, in our ranking formulation we model the user-item-explanation relations for both item and explanation ranking. There is a previous work [43] that similarly considers user-item-aspect relations as a tripartite graph, where aspects are extracted from user reviews. Another branch of related work is tag recommendation for folksonomy [99, 50], where tags are ranked for each given user-item pair. First, our data are different from theirs, since a single tag/aspect when used as an explanation may not be able to clearly explain an item's specialty, e.g., a single word "food" cannot well describe how good a restaurant's food tastes. Second, in terms of problem setting, our work is different from the preceding two, because they solely rank either items/aspects [43] or tags [99, 50], while besides that we also rank item-explanation pairs as a whole in our proposed joint-ranking framework. Another difference is that we study how semantic features of explanations could help enhance the performance of explanation ranking, while none of them did so.

# Chapter 3

# Context-aware Explanation

## 3.1 Background

Context-aware recommender systems (CARS) [2] have been extensively studied [85, 63, 44, 124, 15, 60], because they can adapt to the needs of users in different contextual scenarios, and therefore provide more accurate recommendations. Meanwhile, explainable recommendation, which aims to answer why a particular item is recommended, has drawn increasing attention in recent years [133, 43, 13, 81, 14, 117, 116, 119, 36, 24, 118, 64, 62]. However, few recommendation models have linked the two branches of work for providing context-aware explanation. For instance, one popularly used explanation "You might be interested in [feature], on which this product performs well" [133, 110, 36] does not reflect the recommendation's suitability for users' changing needs under different contexts, which however is especially important when users look for a service-oriented product, e.g., hotel and restaurant.

To produce informative explanations which fit user context (that refers to "any information that can be used to characterize the situation of an entity" [1], such as companion and destination), we propose a novel neural model, called CAESAR standing for Context-Aware Explanation based on Supervised Attention for Recommendations. It is able to select the user's most concerned context as well as its most relevant features to produce context-aware feature-level explanation for

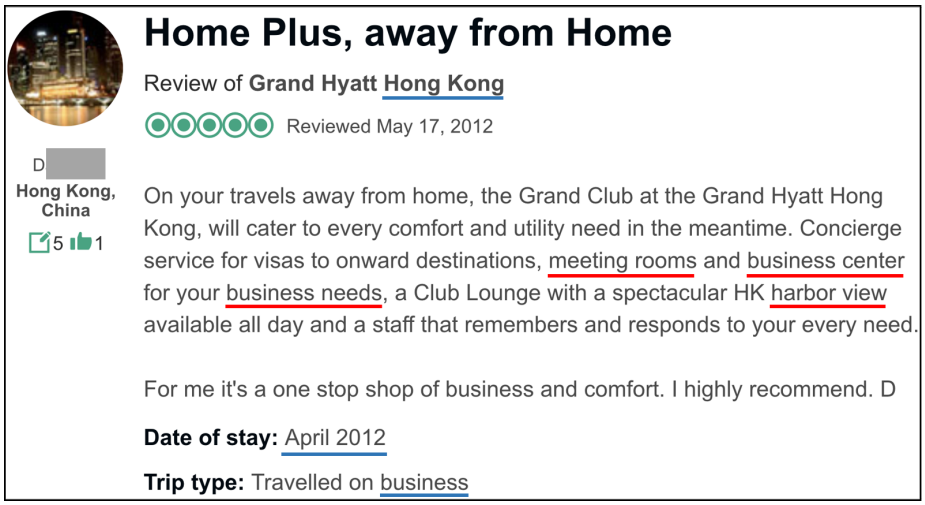



Figure 3.1: A hotel review example from TripAdvisor, where "contexts" are highlighted with blue lines and "contextual features" (i.e., features relevant to the context) with red lines. To protect privacy, the user name is shaded.

the recommendation, i.e., "This product is recommended to you, because its [features] are suitable for your current [context]." The explanation template contains two slots to be filled in by selected features and context, which to some extent is consistent with the aforementioned feature-based explainable recommendation approaches [133, 110, 36]. With these templates, researchers can focus on the design of explanation models.

To produce such context-aware explanation, we need to resolve several challenging issues. The first issue is how to mine high-quality contextual features from user reviews, like the ones highlighted with red lines in Fig. 3.1. For example, pre-defining some contextual seed words and searching for their synonyms [15] may limit the diversity of the extracted features. The second challenging issue is about how to find relevant features for different contexts, because users may care about different features under different situations. For instance, users having a business trip may choose a hotel that provides facilities such as "Wi-Fi" and "meeting room" (see Fig. 3.1), while couples may prefer romantic dinner. Third, how to find a user's most concerned context is also challenging, since not all of the contexts might be equally important to her/him. For example, still in Fig. 3.1, the user only cared about features related to trip goal ("business") and location ("Hong Kong"), but not

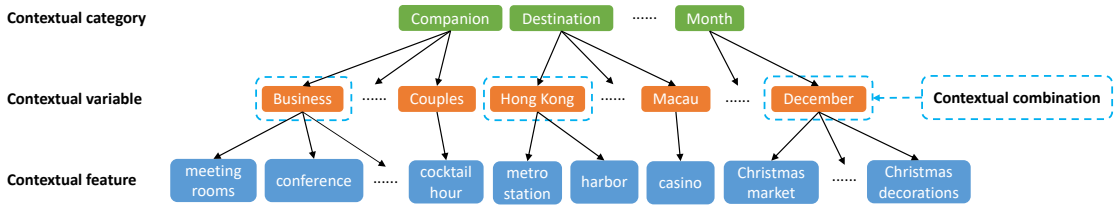



Figure 3.2: Context-related concepts and corresponding examples.

to time. As for related work on CARS, though the popularly applied approach based on tensor factorization [51, 116] can unify users, items, and contexts into one model, it is incapable of modeling contextual features, since that would increase the dimension of tensor and make it highly sparse. Last but not least, it is essential to guarantee the selected features to match to the user's own preference so that the explanation can be personalized.

In this chapter, we present CAESAR to address these limitations. To the best of our knowledge, this is the first work that has explicitly considered different types of user contexts (e.g., companion and destination) to generate feature-level explanation, as well as improving the recommendation accuracy at the same time.

The remaining content of this chapter is organized as follows. We first formulate our problem in Section 3.2, and in detail present the proposed method in Sections 3.3 and 3.4. Then, the experiments and results are discussed in Sections 3.5 and 3.6. We make the final summary in Section 3.7.

## 3.2 Problem Formulation

Before stating the research problem, we introduce some important concepts that will be used in the following content. As shown in Fig. 3.2, contexts can be grouped into different categories that we call *contextual categories*, such as companion, destination, and month. We denote them as $\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_m$, where $m$ is the total number of contextual categories. Each contextual category consists of multiple values, e.g., family and couples for companion. Since a user-item pair normally has one unique value for a contextual category, this *contextual variable* for $\mathcal{C}_j$ (where

$j \in \{1, 2, ..., m\}$) is denoted as $c_j$ for brevity. Following [85], we name the combination of contextual variables $(c_1, c_2, ..., c_m)$ for a user-item pair as *contextual combination* $\mathbf{c}$. In this chapter, we interchangeably call $\mathbf{c}$ *contexts*. As illustrated in Fig. 3.2, there are some features that are highly relevant to a contextual variable, e.g., "meeting rooms" and "conference" to business. We term such features *contextual features* and denote their collection for a contextual variable $c$ from $\mathcal{C}_j$ as $\mathcal{F}_j^c$.

The goal of our recommendation task is to predict a rating $\hat{r}_{u,i,\mathbf{c}}$, given a user $u$, an item $i$, and the corresponding contexts $\mathbf{c}$. Moreover, our proposed model will select contextual features relevant to the user's most concerned context for explanation. At the training stage, the training data consist of users, items, contexts, and contextual features. The key notations and concepts used in this chapter are presented in Table 3.1. We use $\mathbf{p}_u \in \mathbb{R}^d$ to represent the embedding of user $u \in \mathcal{U}$, and $\mathbf{q}_i \in \mathbb{R}^d$ for the embedding of item $i \in \mathcal{I}$. For the $j$-th context in contextual combination $\mathbf{c}$, i.e., $c_j$, we denote its embedding as $\mathbf{k}_j \in \mathbb{R}^d$, and the embedding matrix of contextual features related to this context $\mathbf{H}_j \in \mathbb{R}^{d \times n}$, where $d$ and $n$ respectively represent the dimension of embedding and the number of contextual features.

In Section 3.3, we first introduce our approach to mine contextual features from user reviews, and then describe our proposed model in Section 3.4 that is developed to achieve both context-aware recommendation and explanation.

## 3.3 Contextual Feature Mining

Since some features may not be relevant to certain contextual variables (e.g., feature "hotel" may be too general to context business), it is necessary to find context-relevant features of each contextual variable for producing better explanation. Our approach for mining those features is comprised of two major steps: extracting features from user reviews, and measuring the relevance between features and contexts. For the former, a sentiment analysis toolkit[1] [134] is applied to accomplish it. However, the second step is challenging, as it is not intuitive to assign each extracted

[1] https://github.com/evison/Sentires

Table 3.1: Key notations and concepts.

| Symbol | Description |
|---|---|
| $\mathcal{T}$ | training set |
| $\mathcal{U}$ | set of users |
| $\mathcal{I}$ | set of items |
| $\mathcal{C}_j$ | set of values for the $j$-th contextual category |
| $\mathcal{F}_j^c$ | set of features for context $c$ in category $j$ |
| $\mathbf{p}_u$ | embedding of user $u$ |
| $\mathbf{q}_i$ | embedding of item $i$ |
| $\mathbf{c}$ | contextual combination |
| $\mathbf{k}_j$ | embedding of the $j$-th context in $\mathbf{c}$ |
| $\mathbf{H}_j$ | embedding matrix of features for the $j$-th context in $\mathbf{c}$ |
| $\mathbf{s}_j$ | distribution of features for the $j$-th context in $\mathbf{c}$ |
| $\mathbf{W}$ | weight matrix |
| $\mathbf{w}, \mathbf{b}$ | weight vector |
| $w, b$ | weight scalar |
| $m$ | number of contextual category |
| $n$ | number of contextual feature |
| $d$ | dimension of embedding |
| $r_{u,i,\mathbf{c}}$ | rating assigned by user $u$ on item $i$ under contexts $\mathbf{c}$ |
| $\hat{r}_{u,i,\mathbf{c}}$ | predicted rating |
| $\sigma(\cdot)$ | activation function |

feature an appropriate weight to indicate its relevance to a particular context. To address this challenge, we have revised the weight assigning strategy originally proposed in [60] to identify high-quality contextual features. Specifically, we utilize point-wise mutual information (PMI), instead of raw occurrence frequency in [60], in order to distinguish the relative importance weights of a feature with respect to different contexts. We then compute the weight of a feature by comparing its relevance degrees to different contexts in the same contextual category, which is also different from [60] as it just computes the weight for all contexts regardless of their categories. For example, they treat family and December equally without considering their respective categories companion and month.

To be more concrete, we first extract a list of (feature, opinion, sentiment polarity) entries from textual reviews via the sentiment analysis tool [134], e.g., (harbor view, spectacular, +1) from the sentence "a spectacular HK harbor view available all day". We then filter out infrequently occurring features, as well as negative features because their occurrence is much less than that of positive features, which may cause data imbalance issue if we integrate them. Next, we aim at discovering the most relevant features to each context associated with the target review by analyzing the relation between the features and the context. To this end, we first count a feature $f$'s overall occurrence frequency $freq_f^c$ in the user reviews pertaining to context $c$ in the contextual category $\mathcal{C}_j$, where $j \in \{1, 2, ..., m\}$. To account for the relative importance of a feature to different contexts, we compute the PMI value:

$$PMI_f^c = \frac{freq_f^c}{freq_f \cdot freq^c} \tag{3.3.1}$$

where $freq_f$ denotes the total frequency of feature $f$ in all the user reviews and $freq^c$ indicates the total number of features in the user reviews pertinent to context $c$.

With the PMI values, we calculate the mean value for each feature $f$ over all contexts in $\mathcal{C}_j$ as $avg_f$, based on which we estimate the statistical error of feature $f$ for context $c$ as $err_f^c$, whose absolute value can be seen as the relevance degree

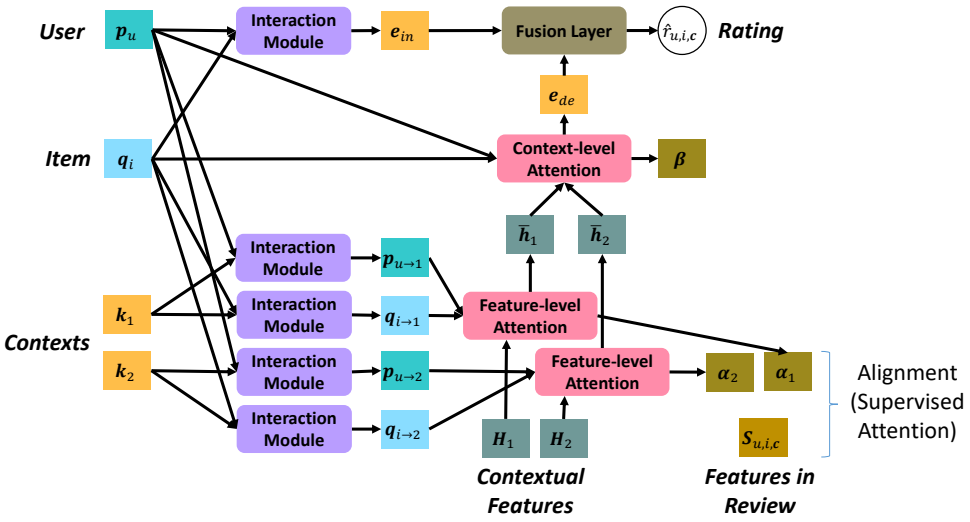



Figure 3.3: Overview of our proposed model CAESAR, where $\mathbf{p}_u$, $\mathbf{q}_i$, $\mathbf{k}_1$, $\mathbf{k}_2$, $\mathbf{H}_1$, $\mathbf{H}_2$, and $\mathbf{S}_{u,i,\mathbf{c}}$ are input embeddings, and $\hat{r}_{u,i,\mathbf{c}}$ is the final predicted rating. $\mathbf{S}_{u,i,\mathbf{c}}$ corresponds to the ground-truth distribution of context-aware features in the user review.

between the feature and the context:

$$\begin{aligned} avg_f &= \frac{1}{|\mathcal{C}_j|} \sum_{c \in \mathcal{C}_j} PMI_f^c \\ err_f^c &= PMI_f^c - avg_f \\ w_f^c &= \left| err_f^c \right| \end{aligned} \tag{3.3.2}$$

The larger the weight $w_f^c$ is, the less general the feature $f$ is to the context $c$, indicating it is more relevant to this context. For each context, we rank all the features according to their weights. Moreover, the ranking positions of features that have the same weight for a certain context are adjusted in accordance with their appearing frequencies under this context.

Lastly, for each context $c$ in the contextual category $\mathcal{C}_j$, we select top $n$ ranked features to construct the contextual feature set $\mathcal{F}_j^c$.

## 3.4 Model Description

The results of the previous section are leveraged to generate context-aware recommendation and explanation. In the following, we introduce our proposed model

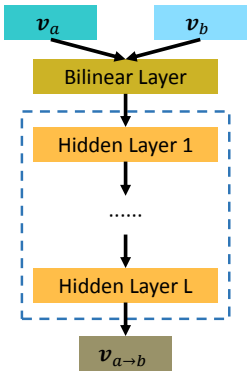



Figure 3.4: Interaction module.

CAESAR (standing for Context-Aware Explanation based on Supervised Attention for Recommendations), where we design a *two-level attention* mechanism in order to discriminate the importance of different contexts as well as their associated features. In addition, we propose a *supervised attention* mechanism to align contextual features with those in the target review, so that the selected features for explanation can match to the user's preference.

### 3.4.1 Model Basics

As shown in Fig. 3.3, the interaction module and the attention module are integrated into our model, so we first briefly introduce these two basic building blocks.

**Interaction Module.** We propose to employ multi-layer perceptron [85, 63] to learn the non-linear interaction between two entities among users, items, and contexts, since it has been demonstrated with better representation ability [45] than linear interaction such as matrix factorization [88]. For the convenience of later usage, we write this module as:

$$\mathbf{v}_{a\to b} = Inter(\mathbf{v}_a, \mathbf{v}_b) \tag{3.4.3}$$

where $\mathbf{v}_a \in \mathbb{R}^d$ and $\mathbf{v}_b \in \mathbb{R}^d$ are input vectors, $Inter(\cdot)$ denotes the function of interaction module, and $\mathbf{v}_{a\to b} \in \mathbb{R}^d$ is the output vector. As shown in Fig. 3.4, we first map the embeddings of two entities $\mathbf{v}_a$ and $\mathbf{v}_b$ to a shared hidden space via a bilinear layer. Then the resultant vector is fed into a stack of fully connected

layers to enable non-linear transformations. Finally, the output vector from the last hidden layer is transformed into $\mathbf{v}_{a\to b}$, so that it has the same dimension as the input vectors. More specifically, the interaction module in our model is formally defined as:

$$\begin{aligned}
\mathbf{z}_0 &= \sigma(\mathbf{W}_0[\mathbf{v}_a, \mathbf{v}_b] + \mathbf{b}_0) \\
\mathbf{z}_1 &= \sigma(\mathbf{W}_1\mathbf{z}_0 + \mathbf{b}_1) \\
&\cdots\cdots \\
\mathbf{z}_L &= \sigma(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L) \\
\mathbf{v}_{a\to b} &= \mathbf{W}_{L+1}\mathbf{z}_L + \mathbf{b}_{L+1}
\end{aligned} \tag{3.4.4}$$

where $[\cdot,\cdot]$ denotes the concatenation of vectors, $\sigma(\cdot)$ is a non-linear activation function, $\mathbf{W}_x \in \mathbb{R}^{2d\times 2d}$ and $\mathbf{b}_x \in \mathbb{R}^{2d}$ are respectively the weight matrix and bias vector in the hidden layers, and $\mathbf{W}_{L+1} \in \mathbb{R}^{d\times 2d}$ and $\mathbf{b}_{L+1} \in \mathbb{R}^d$ correspond to the weight and bias in the final linear layer.

**Attention Module.** The attention mechanism [105, 14, 18, 80, 124] is employed on contexts and contextual features, because users under different contextual situations may have different needs, and different contexts may have different impacts. We formally denote the attention module as:

$$\overline{\mathbf{v}}, \boldsymbol{\alpha} = Attn(\mathbf{V}, \mathbf{v}_a, \mathbf{v}_b) \tag{3.4.5}$$

where $\mathbf{v}_a \in \mathbb{R}^d$ and $\mathbf{v}_b \in \mathbb{R}^d$ are the input vectors, $\mathbf{V} \in \mathbb{R}^{d\times z}$ represents the input matrix, $Attn(\cdot)$ denotes the function of attention module, $\overline{\mathbf{v}} \in \mathbb{R}^d$ is the aggregated output vector, and $\boldsymbol{\alpha} \in \mathbb{R}^z$ is the output vector consisting of attention scores. As illustrated in Fig. 3.5, each object $\mathbf{v}_l \in \mathbf{V}$ is fed into the attention network together with input vectors $\mathbf{v}_a$ and $\mathbf{v}_b$ for computing a weight score. Formally, we define the attention network as:

$$\begin{aligned}
\alpha_l^* &= \mathbf{w}^\top \sigma(\mathbf{W}[\mathbf{v}_l, \mathbf{v}_a, \mathbf{v}_b] + \mathbf{b}) + b \\
\alpha_l &= \frac{\exp(\alpha_l^*)}{\sum_{l'=1}^{z} \exp(\alpha_{l'}^*)}
\end{aligned} \tag{3.4.6}$$

where $\mathbf{W} \in \mathbb{R}^{3d\times 3d}$, $\mathbf{b} \in \mathbb{R}^{3d}$, $\mathbf{w} \in \mathbb{R}^{3d}$, and $b \in \mathbb{R}$ are model parameters. $\boldsymbol{\alpha}^*$ is normalized through the softmax function, resulting in the final attention vector $\boldsymbol{\alpha}$,

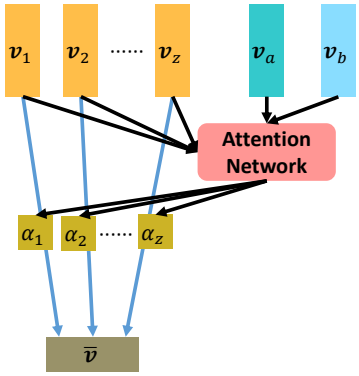



Figure 3.5: Attention module.

with which we calculate the weighted sum $\overline{\mathbf{v}} = \sum_{l'=1}^{z} \alpha_{l'} \mathbf{v}_{l'}$ over the input embedding matrix $\mathbf{V}$.

### 3.4.2 Personalized Recommendation

As shown in Fig. 3.3, given a user $u$, an item $i$, and the contextual combination $\mathbf{c}$, we first retrieve their corresponding embeddings $\mathbf{p}_u$, $\mathbf{q}_i$, and $\mathbf{k}_j$ for each context $c_j$ in $\mathbf{c}$, where $j \in \{1, 2, ..., m\}$. To model the effects of contexts on a user, we employ the interaction module on the embeddings of user $u$ and each context $c_j$ via Eq. (3.4.3), $\mathbf{p}_{u \to j} = Inter(\mathbf{p}_u, \mathbf{k}_j)$. In a similar way, we obtain the item's contextual embedding $\mathbf{q}_{i \to j}$.

As the contextual embeddings $\mathbf{p}_{u \to j}$ and $\mathbf{q}_{i \to j}$ are latent, we propose to characterize users' preferences over contextual features. For this purpose, with the contextual embeddings $\mathbf{p}_{u \to j}$ and $\mathbf{q}_{i \to j}$, and the corresponding embedding matrix of contextual features $\mathbf{H}_j$ for context $c_j$, we obtain the overall assessment of these features $\overline{\mathbf{h}}_j$ and their attention scores $\boldsymbol{\alpha}_j$ using Eq. (3.4.5), i.e., $\overline{\mathbf{h}}_j, \boldsymbol{\alpha}_j = Attn(\mathbf{H}_j, \mathbf{p}_{u \to j}, \mathbf{q}_{i \to j})$. Notice that we apply different attention modules to contexts in $\mathbf{c}$, and call them *feature-level attention* as a whole. In this way, different contextual features' importance to a certain context can be individually learned.

To further distinguish the importance of different contexts to the target user and item, we represent the embeddings from feature-level attention component as a

matrix:

$$\overline{\mathbf{H}} = \begin{bmatrix} | & & | & & | \\ \overline{\mathbf{h}}_1 & \dots & \overline{\mathbf{h}}_j & \dots & \overline{\mathbf{h}}_m \\ | & & | & & | \end{bmatrix}$$

where $\overline{\mathbf{H}} \in \mathbb{R}^{d \times m}$ and $\overline{\mathbf{h}}_j \in \mathbb{R}^d$. Then, the influences of different contexts can be modeled via another attention module, i.e., $\mathbf{e}_{de}, \boldsymbol{\beta} = Attn(\overline{\mathbf{H}}, \mathbf{p}_u, \mathbf{q}_i)$, which we call *context-level attention*. Here $\mathbf{e}_{de}$ is context-dependent preference because it is inferred from contextual data including contexts and contextual features. On the other hand, there may exist another type of preference, i.e., context-independent preference, which remains stable regardless of the context [15]. To model it, we employ another interaction module without contexts involved, i.e., $\mathbf{e}_{in} = Inter(\mathbf{p}_u, \mathbf{q}_i)$.

We finally pass both types of preference into a fusion layer, by which the rating can be predicted as:

$$\hat{r}_{u,i,\mathbf{c}} = \mathbf{w}_r^\top [\mathbf{e}_{in}, \mathbf{e}_{de}] + b_r \tag{3.4.7}$$

where $\mathbf{w}_r \in \mathbb{R}^{2d}$ and $b_r \in \mathbb{R}$. For the rating prediction task, we adopt the mean squared error loss function:

$$\mathcal{L}_r = \sum_{(u,i,\mathbf{c}) \in \mathcal{T}} (r_{u,i,\mathbf{c}} - \hat{r}_{u,i,\mathbf{c}})^2 \tag{3.4.8}$$

where $\mathcal{T}$ is the training set and $r_{u,i,\mathbf{c}}$ denotes the ground-truth rating that the user $u$ assigned to item $i$ under contexts $\mathbf{c}$.

### 3.4.3 Explanation Generation

The embedding matrix $\mathbf{H}_j$ leverages contextual features in the feature-level attention component, but they are latent because there is no direct connection with explicit features. To build such connection for generating explanation, we force the scores resulting from feature-level attention to be the same as the ground-truth frequencies of contextual features in the target review.

Concretely, from the target review we can have the occurrence frequency $freq_f^{c_j}$ of each feature $f$ that appears in context $c_j$'s feature collection $\mathcal{F}_j^{c_j}$ (from Section

3.3). We can then obtain the feature distribution vector $\mathbf{s}_j$ (the matrix form is $\mathbf{S}_{u,i,\mathbf{c}}$ as shown in Fig. 3.3), in which each element corresponds to a contextual feature's normalized frequency:

$$s_j^f = freq_f^{c_j} / \sum_{f' \in \mathcal{F}_j^{c_j}} freq_{f'}^{c_j} \tag{3.4.9}$$

To align the attention vector $\boldsymbol{\alpha}_j$ with the distribution vector $\mathbf{s}_j$, we apply the mean squared error loss function:

$$\mathcal{L}_e = \sum_{(u,i,\mathbf{c}) \in \mathcal{T}} \sum_{j=1}^{m} \sum_{k=1}^{n} (s_j^k - \alpha_j^k)^2 \tag{3.4.10}$$

where $m$ is the total number of contexts in $\mathbf{c}$ and $n$ denotes the number of contextual features. As the attention scores are learned from explicit contextual features through the manner of supervised learning, we name it *supervised attention* [78]. To explain the recommendation, we select a context with the highest score from context-level attention and its most relevant contextual features from feature-level attention to fill in the template: "This product is recommended to you, because its [features] are suitable for your current [context]."

### 3.4.4 Multi-task Learning

At last, we integrate the rating prediction task and the context-aware explanation task into a unified multi-task learning framework, for which the objective function is:

$$\mathcal{J} = \min_{\Theta} (\lambda_r \mathcal{L}_r + \lambda_e \mathcal{L}_e + \lambda_n ||\Theta||_F^2) \tag{3.4.11}$$

where $\Theta$ is the set of model parameters, and $\lambda_r$, $\lambda_e$ and $\lambda_n$ are regularization coefficients for different tasks.

## 3.5 Experimental Setup

In this section, we present our experimental setup, including dataset description, evaluation metrics, baseline models and implementation details. We also study some important hyper-parameters of our model CAESAR.

Table 3.2: Statistics of our datasets.

| | **TripAdvisor** | **Yelp** |
|---|---|---|
| # of users | 9,765 | 27,147 |
| # of items | 6,280 | 20,266 |
| # of reviews | 320,023 | 1,293,247 |
| Avg. # of reviews / user | 32.77 | 47.64 |
| Avg. # of reviews / item | 50.96 | 63.81 |
| # of contextual variables in companion | 6 | - |
| # of contextual variables in day of a week | - | 7 |
| # of contextual variables in month | 13 | 12 |
| # of contextual variables in destination | 415 | 242 |

### 3.5.1 Datasets

In our experiment, we used two large-scale datasets from two typical service domains, i.e., hotel and restaurant, to evaluate our proposed model. For hotel domain, we constructed the dataset with reviews collected from a popular travel website TripAdvisor[2]. Specifically, we crawled all the user reviews to every hotel located in Hong Kong from this website, as well as those users' historical reviews in other cities. After removing non-English reviews, we have in total 2,118,108 interaction records. The other dataset is from Yelp Challenge 2019[3], which contains 6,685,900 restaurant reviews. Since the two datasets are very large and cold start problem is not our focus, we further processed them by recursively removing users and items with less than 20 interactions, which results in two subsets (see Table 3.2 for their descriptive characteristics).

Each review record in our datasets comprises user ID, item ID, overall rating in the scale of 1 to 5, textual review, and contexts in which the user was experiencing that item. As indicated in [85], taking more contextual categories into account

[2]https://www.tripadvisor.com

[3]https://www.yelp.com/dataset/challenge

could lead to better performance, so we make use of all the available contexts associated with each review. Specifically, the contextual categories consist of companion, month, and destination for TripAdvisor dataset, and day of a week, month, and destination for Yelp. Notice that the contextual categories companion and month in TripAdvisor indicate with whom and in which month a user stayed in a hotel, but Yelp does not provide the exact time a user was dining in a restaurant, so the contextual variables for day of a week and month are inferred from the review time. For the contextual category destination in both datasets, we took the target city (where the item is located) as the contextual variable.

### 3.5.2 Evaluation Metrics

To compare the recommendation performance of different methods, we adopt two commonly used metrics in recommender systems: Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).

RMSE is calculated by estimating the quadratic difference between ground-truth rating $r_{u,i,\mathbf{c}}$ and the predicted one $\hat{r}_{u,i,\mathbf{c}}$:

$$RMSE = \sqrt{\frac{1}{N}\sum_{u,i,\mathbf{c}}(r_{u,i,\mathbf{c}} - \hat{r}_{u,i,\mathbf{c}})^2} \tag{3.5.12}$$

where $N$ is the number of instances in the testing set.

Similarly, MAE can be calculated via the following formula:

$$MAE = \frac{1}{N}\sum_{u,i,\mathbf{c}}|r_{u,i,\mathbf{c}} - \hat{r}_{u,i,\mathbf{c}}| \tag{3.5.13}$$

For both metrics, a lower value indicates a better performance.

As to the evaluation of context-aware explainability, we conduct a small-scale user survey to evaluate the explanations, because there is no available metrics.

### 3.5.3 Compared Methods

To evaluate the performance of our CAESAR, we compare it with the following state-of-the-art methods:

- **PMF**: Probabilistic Matrix Factorization [88]. This is the standard matrix factorization method that characterizes users and items by latent factors inferred from observed ratings. We take it as the context-unaware baseline. Its objective function is optimized by alternative least square (ALS).

- **EFM**: Explicit Factor Models [133]. It is a joint matrix factorization model in which user-feature attention and item-feature quality are considered for explaining recommendations. It is the feature-level explanation baseline, without considering user contexts. Stochastic gradient descent (SGD) is introduced to optimize its objective function in our implementation.

- **AFM**: Attentional Factorization Machines [124]. This model extends factorization machines (FM) [97] by learning the importance of each feature interaction via a neural attention network. It is a context-aware approach, but treats users, items, and contexts equally as sparse features.

- **NFM**: Neural Factorization Machines [44]. It is a more generalized FM built upon neural network for learning high-order feature interactions in a nonlinear way for dealing with sparse data. The way it models contexts is similar to AFM.

- **AIN**: Attentive Interaction Network [85]. To model the effects of contexts on users and items, this model employs two pathways of interaction module, followed by attention mechanism to discriminate the impacts of different contexts on users and items.

We omit the comparison with other context-aware models such as Multiverse [51], FM [97], CAMF [6] and COT [77], since they are outperformed by the recently proposed AIN as shown in [85].

### 3.5.4 Implementation Details

We randomly divide each dataset into training (80%), validation (10%), and testing (10%) sets, and guarantee that each user/item has at least one instance in the

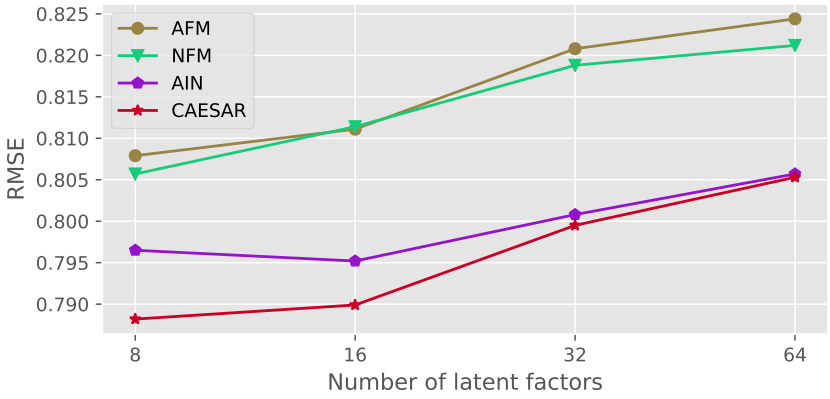



Figure 3.6: RMSE on TripAdvisor with varied numbers of latent factors in four compared context-aware methods.

training set. We repeat the splitting process for 5 times. The validation set is used for hyper-parameters tuning, and we report the average performance on the testing set. The early stopping strategy is performed on all the models, i.e., a model's RMSE and MAE on the testing set are reported when it reaches the best performance on the validation set.

We implemented AIN and our CAESAR in Python using TensorFlow, and adopted the codes of AFM and NFM shared by their authors. All the neural network based methods, i.e., AFM, NFM, AIN and CAESAR, are optimized by Adam [55]. The initial learning rate of AFM and NFM is searched in [0.005, 0.01, 0.02, 0.05], following the setting in [44]. The number of latent factors of these models is searched in [8, 16, 32, 64]. Fig. 3.6 shows the recommendation accuracy by varying the value of this parameter on TripAdvisor dataset[4]. From the figure, we can see that these methods generally share the similar trend: fewer latent factors (e.g., 8 for AFM, NFM and CAESAR, and 16 for AIN) lead to better accuracy, while increasing the number of latent factors deteriorates the performance, because too many latent factors may cause overfitting. Therefore, on TripAdvisor we set the number of latent factors of AFM, NFM and CAESAR to 8, and that of AIN to 16. Matrix factorization methods PMF and EFM were also implemented in Python. For PMF, we set the dimension of latent factors to 20 following [52], and search trade-off parameters

[4]The results on Yelp dataset are similar, so we do not show here.

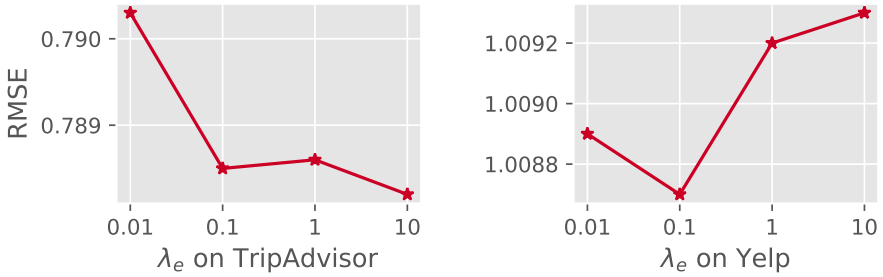
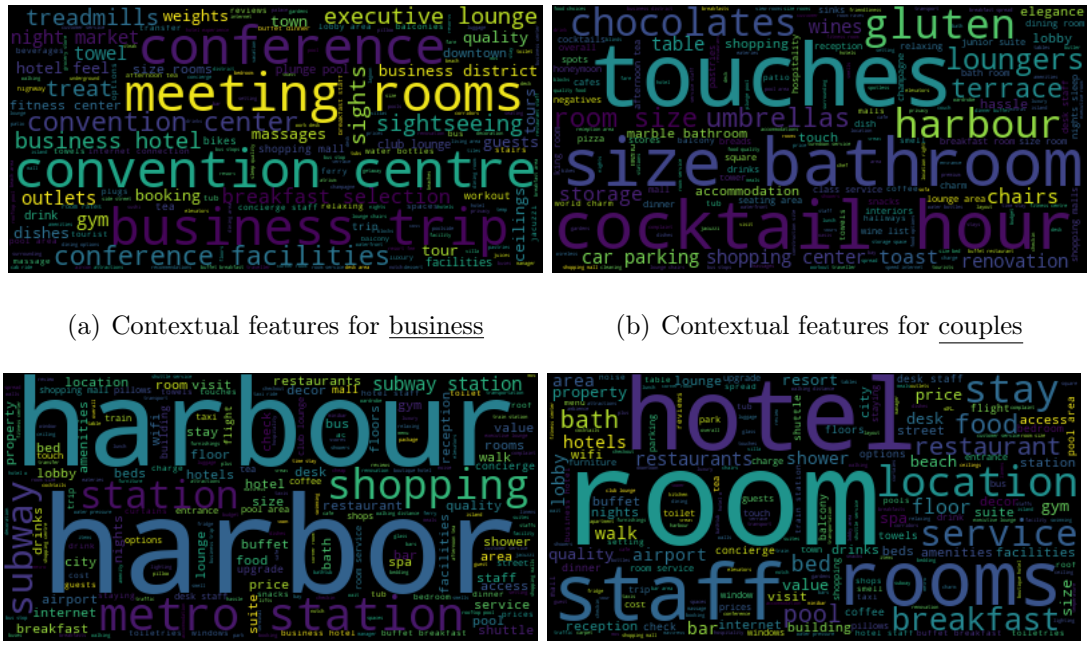



Figure 3.7: RMSE on TripAdvisor (left) and Yelp (right) with different regularization coefficients on $\lambda_e$ in our model CAESAR.

in [0.1, 1, 10, 100]. For EFM, the numbers of explicit factors and implicit factors are set equal and both searched in [8, 16, 32, 64]. We reuse the other hyper-parameters of the baselines as reported in the original papers. Furthermore, the weight and bias parameters of all the methods are learned from scratch with random initialization for fair comparison.

For our model CAESAR, the learning rate is set to 0.0001, the batch size is 64, and the number of contextual features $n$ for each context is 100. We use $ReLU(\cdot)$ as the activation function, and employ 2 hidden layers for interaction module in the condition of modeling context-independent preference and 1 hidden layer for context-dependent preference. For the regularization coefficients on different tasks, we set $\lambda_r$ to 1 and $\lambda_n$ to 0.0001, and search $\lambda_e$ in [0.01, 0.1, 1, 10]. The left part of Fig. 3.7 shows our method's performance w.r.t. $\lambda_e$ on TripAdvisor dataset and the right part shows that on Yelp. As it can be seen, the optimal values of $\lambda_e$ on two datasets are different, i.e., 10 for TripAdvisor and 0.1 for Yelp. The larger the regularization coefficient $\lambda_e$ is, the closer the scores from feature-level attention are to the distribution of contextual features in user reviews. Notice that since the contextual variables for day of a week and month in Yelp are not the exact time when a user was dining, the inferred features may not precisely match those expressed in the target review, which may cause difference between the attention scores and the distribution vector of contextual features (i.e., small $\lambda_e$).




(a) Contextual features for business

(b) Contextual features for couples

(c) Contextual features for Hong Kong

(d) Features according to occurring frequency

Figure 3.8: Word clouds of features identified by our contextual feature mining approach (a, b, and c) and that based on occurring frequency (d) on TripAdvisor dataset.

## 3.6 Results and Analysis

To study our model's explainability, we provide analysis on features mined by our contextual feature mining approach, and human evaluation and case study on explanations as generated by CAESAR. We also present the recommendation results in comparison with baseline models.

### 3.6.1 Contextual Feature Analysis

We generate word clouds for features on TripAdvisor dataset (see Fig. 3.8), as mined by our contextual feature mining approach (the results on Yelp dataset show similar pattern). The size of features in sub-figures (a), (b) and (c) indicates the weight score computed via Eq. (3.3.2), and that in (d) represents a feature's occurring frequency in all the user reviews.

It can be seen that, under the contextual situation business, users have paid

more attention to facilities like "meeting rooms", "conference" and "convention centre". Relatively, users under couples have preferred leisurely and romantic features, such as "cocktail hour" and "chocolates". For the contextual category destination, our approach is able to find features that reveal a place's characteristics, e.g., "harbor", "shopping", and "metro station" in Hong Kong. It hence shows that our contextual feature mining approach is capable of discovering context-aware features. In comparison, the features identified according to occurring frequency in Fig. 3.8 (d), which have been commonly adopted by existing explainable recommendation approaches [133, 116], primarily reflect some general aspects such as "hotel", "room" and "staff", which are not context-specific.

### 3.6.2 Human Evaluation on Explanations

To evaluate the quality of explanations produced by our method CAESAR, we conduct a human evaluation on TripAdvisor, because the contexts on this dataset are more precise than that of Yelp as discussed in Section 3.5.4. Compared to automatic evaluation with offline metrics, the results of human evaluation could be more convincing and reliable, since they reflect real people's actual perception towards explanations. However, large-scale human evaluation could be very expensive, so automatic evaluation is preferred by researchers given that it is more convenient and easy-to-implement. With these considerations, we opt for a small-scale human evaluation.

Specifically, in our questionnaire, we prepared two questions, each containing 10 different cases. We then invited 10 people to answer those questions. Specifically, they all are Chinese, and currently doing either Ph.D. or M.Phil. in computer science, so their English language proficiency is qualified for this evaluation. Their gender distribution is well balanced (5 males vs. 5 females), and their ages are distributed between 23 and 30. As a consequence of the demographics, their evaluation may only reflect a small group of people's perception to the explanations. For example, American with high school diploma may value the explanations differ-

ently. Moreover, since the participants are experts in the field of computer science, their domain knowledge may cause certain bias to the evaluation. For example, it could be easier for them than ordinary people to understand what *context* means, so their evaluation could be biased to more detailed explanations. Nevertheless, we conducted this small-scale human evaluation, in order to quickly obtain some initial results.

The first question (Q1) aims at studying which type of explanations, either context-aware or context-unaware, could be more helpful. For each case we provided two explanations respectively returned by our method CAESAR and the baseline EFM [133]. Specifically, the top 5 features as selected by each method are filled in its explanation template as shown in Table 3.3. In addition, our method adds one context related to those features. Along with the explanation and its corresponding recommendation (a hotel), we also provided the user's contexts and her/his review to the hotel. The human judges were then asked to indicate which explanation is more helpful for them to understand the recommendation.

The second question (Q2) was designed to investigate whether the features given in the explanation could well describe the selected context. Considering the fact that there is no available context-aware explanation method to compare at the time of our experiment, we adopted two simple baselines: RAND and POP. In more details, given a context and its context-aware features obtained from Section 3.3, RAND randomly selects some context-aware features, while POP selects features based on its occurring frequency. Notice that, the details of these three methods were hidden to human judges and the feature lists were randomly shuffled, which was to fairly compare their performance. Then, the human judges were asked to evaluate which feature list is more suitable to describe the given contextual situation.

Specifically, the two evaluation questions, each with one example case, are displayed below.

- **Q1**: After reading the given information, do you think which explanation is more helpful for you to understand the recommendation?

**Hotel**: Novotel Citygate Hong Kong

**Context**: travelled to Hong Kong as a Couple on February

**Review**: Hotel is a short distance from the airport via free shuttle every 15 mins. There is an MTR station and bus stop at the base of the Citigate Mall and the Ferry and Cable Car is a short walk from the hotel. Disneyland is just up the road and the Giant Buddha at the top of the hill (Cable Car, bus or taxi). So the Citigate offers more touring options than a typical airport hotel. The hotel meets the typical Accor expectations ...

- A. This hotel is recommended to you, because its features [symphony, mtr station, mongkok, bldg, airport access] are suitable for your current context [Hong Kong].
- B. You might be interested in features [room, hotel, staff, rooms, location], on which this hotel performs well.

- **Q2**: Imagine you are under a specific context when booking a hotel, which feature list do you think better describes this context?

**Context**: December

A. floor, selections, apartment, queue, service

B. Christmas market, premier rooms, birthday stay, flyover, island side

C. service, floor, area, walk, property

We keep EFM's original template format, as it has been widely adopted in related work [36, 116, 133], and design our own one which is more suitable for context-aware explanation. The template difference could potentially influence the response of participants, but we believe that the highlighted features/contexts are what they concern. The evaluation results are depicted in Fig. 3.9, where the bar charts show the vote numbers on the 10 testing cases for different methods, and the pie charts

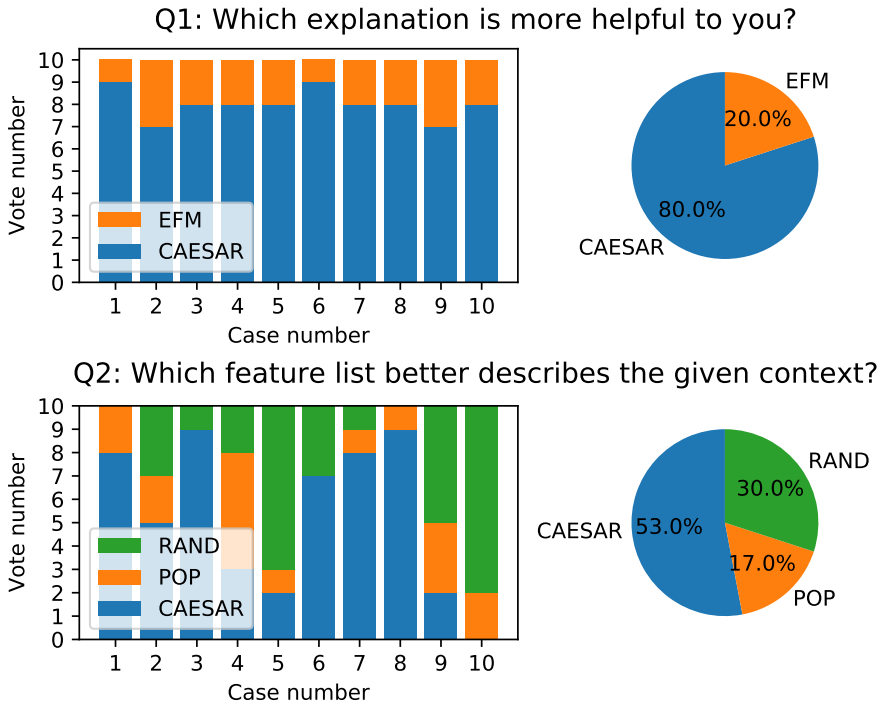



Figure 3.9: Results of human evaluation on explanations provided by compared methods on TripAdvisor dataset. The bar charts show the exact vote numbers on different testing cases, and the pie charts illustrate the voting percentages. For EFM in Q1 and POP in Q2, CAESAR is significantly better than them with $p < 0.01$ via Student's t-test.

show the percentages of votes.

For Q1, regarding all of the cases, most human judges regard the explanations generated by our method CAESAR being more helpful than EFM for them to understand the recommendation. Moreover, the results are statistically significant. This validates our model's capability of returning more useful explanations. As to Q2, our method obtains 53% votes, which is obviously higher than those of the two baselines (30% to RAND and 17% to POP). This confirms that to a large extent our method can find features that better characterize a context. From the bar chart, we also observe that our method gains more than 5 votes on 6 different cases, while RAND dominates on 3 cases, leaving POP on only one case. This might be because RAND and POP both leverage the context-aware features as obtained through our feature mining approach (see Section 3.3), which may enable them to return some

reasonable features in some cases. This again demonstrates the effectiveness of our contextual feature mining approach in finding context-aware features.

### 3.6.3 Case Study on Explanations

To qualitatively compare the explanations generated by CAESAR and those by EFM [133], a case study is shown in Table 3.3. The context and feature underlined are our model's selected elements to generate the context-aware explanation; while for EFM only the selected feature is underlined because it is context-unaware.

From the examples we can see that when the same user visits two different hotels under different contexts, our model can adaptively select the most important context to her as well as the most relevant features. More specifically, our model selects "rooftop view", "floor levels", "smorgasbord", and "eating establishments", to match the couples context, and selects "harbor" and "location" for the destination Sydney. In comparison, EFM just selects some general features for the two hotels, such as "room" and "staff", which can not distinguish the user's needs under different contexts.

### 3.6.4 Recommendation Performance

The accuracy of our model CAESAR in comparison with baseline models on two datasets is given in Table 3.4. From this table, we can have several observations.

Firstly, CAESAR consistently achieves the best performance w.r.t. RMSE and MAE on two datasets. It can be attributed to our model structure that incorporates explicit contextual features as extracted from users reviews as an important information source. This illustrates the necessity of fusing explicit features into the context-aware model, and demonstrates the capability of our proposed supervised attention mechanism to deal with these features. In addition, we notice that the performance gap between CAESAR and AIN is relatively small. Our primary goal has been to achieve explainability, for which we proposed the supervised attention mechanism, which aligns the attention weights with the feature distribution in the

Table 3.3: A case study for explanation comparison between our method CAESAR and the baseline EFM when recommending two hotels to the same user. The texts underlined are automatically selected context/feature to fill in the explanation template.

| Item | Contexts | | Features | | Features in Testing Review | **Explanation** | |
|---|---|---|---|---|---|---|---|
| | CAESAR | EFM | CAESAR | EFM | | CAESAR | EFM |
| Hotel Madera Hong Kong | Couples, September, Hong Kong | - | rooftop view, parking bays, floor levels, smorgasbord, eating establishments | room, hotel, staff, stay | bar, rooftop view, hotels, cafe, rooftop, bedroom, balcony, breakfast, location, glass, floors, room, station, hotel | This product is recommended to you, because its [rooftop view] is suitable for your current context [couples]. | You might be interested in [room], on which this product performs well. |
| Meriton Suites Kent Street | Business, October, Sydney | - | harbor, location, apartment, lifts | room, hotel, staff, location | location, windows, noise, rooms | This product is recommended to you, because its [harbor] is suitable for your current context [Sydney]. | You might be interested in [room], on which this product performs well. |

Table 3.4: Performance comparison of all methods in terms of RMSE and MAE. The best performing method and values are boldfaced. * indicates that our model CAESAR performs significantly better than NFM through Student's t-test ($p < 0.01$).

| Model | TripAdvisor | | Yelp | |
|---|---|---|---|---|
| | RMSE | MAE | RMSE | MAE |
| PMF | 0.8703 | 0.6961 | 1.0856 | 0.8765 |
| EFM | 0.8415 | 0.6403 | 1.0557 | 0.8243 |
| AFM | 0.8102 | 0.6181 | 1.0355 | 0.8060 |
| NFM | 0.8095 | 0.6177 | 1.0366 | 0.8056 |
| AIN | 0.7952 | 0.6072 | 1.0111 | 0.7879 |
| **CAESAR** | **0.7898*** | **0.6022*** | **1.0080*** | **0.7814*** |

target user review. This mechanism differs from vanilla attention, so it may sacrifice the accuracy to a certain degree.

Secondly, factorization machines (FM) based models (i.e., AFM and NFM) obtain similar performance, but are dominated by the other context-aware methods (i.e., CAESAR and AIN) on two datasets. Although AFM and NFM enhance FM via neural network, they treat users, items and contexts as sparse features and model all types of interactions in a similar way. In comparison, the other context-aware methods employ two pathways of interaction module to characterize the effects of different contexts on users and items, and subsequently adopt attention mechanism to dynamically infer representations of users and items. This verifies the importance of modeling different types of interactions in different ways, and the respective roles of interaction module and attention module in our model.

Thirdly, context-unaware models, i.e, PMF and EFM, underperform all the other methods that take into account contextual information during recommendation process. This is not surprising, as users are likely to make decision according to their contextual situations especially for service products. As such, context-aware models can better characterize users' preferences over item features. Besides, since PMF

and EFM are both based on matrix factorization, its linearity may make user and item modeling limited [45], in comparison with other neural network based methods that model users and items in a non-linear way. In addition, we observe that the performance of EFM is much better than that of PMF, which may be because the former takes features extracted from user reviews as complementary information into its modeling process. This again demonstrates the usefulness of explicit features to improve recommendation performance.

## 3.7 Summary

In this chapter, we propose a novel neural model called CAESAR, which is experimentally demonstrated to be capable of characterizing users' preferences over contextual features mined from user reviews, for achieving both context-aware recommendation and explanation. In particular, our designed two-level attention mechanism can distinguish the importance of different contexts and their related features. Moreover, the supervised attention can align explicit contextual features with implicit ones to generate context-aware explanation, while still being able to enhance recommendation accuracy simultaneously. Experimental results on two real-world datasets (TripAdvisor and Yelp) show that our model outperforms the state-of-the-art baselines in terms of not only rating prediction accuracy, but also returning context-aware feature-level explanations that are more helpful than context-unaware explanations as judged by human evaluators.

# Chapter 4

# Neural Template Explanation Generation

## 4.1 Background

In the previous chapter, we introduced a context-aware explanation method, which is template-based. A template is a word sequence with some segments acting as its backbones, and the remaining slots are filled in by input text [121, 11]. In many template-based explanation approaches, the template is usually defined as "You might be interested in [feature], on which this product performs well" [133], where the feature(s) are predicted by means of matrix/tensor factorization [133, 116] or attention mechanism [36]. However, manually defined templates are expensive to create. Moreover, they also restrict the expressiveness of explanations. For example, in the above template all item features are described as "performs well", which could not reflect the special property of different features.

In the meantime, natural language generation approaches have obtained research interests recently, owing to their flexibility in text styles. The goal of these approaches is to automatically produce flexible sentences as learned from user-generated content, e.g., user reviews. For instance, Attribute-to-Sequence (Att2Seq) [31], a state-of-the-art review generation method, produces a variety of expressions

Table 4.1: Example explanations generated by state-of-the-art natural language generation methods (Att2Seq [31] and NRT [70]) and our NETE. The reference sentence is the ground-truth explanation extracted from user reviews.

| | |
|---|---|
| Reference | They have a huge **variety** of things. |
| NRT | The food is good. |
| Att2Seq | I'm not sure if I need to go back. |
| **NETE** | They have a **variety** of things to choose from. |
| Reference | The black garlic **ramen** was good as well. |
| NRT | The food is good. |
| Att2Seq | The food was great. |
| **NETE** | The **ramen** was delicious. |

(see the examples in Table 4.1). Another typical method, Neural Rating and Tips generation (NRT) [70], aims to generate short and concise tips. Despite of that, there are two important issues yet to be addressed in current natural language generation approaches. First, since these approaches are trained on user-generated content whose quality cannot always be guaranteed, the topic of the generated sentences may be irrelevant to the recommended item, e.g., "I'm not sure if I need to go back". Second, due to the lack of variety in generative signals, a large proportion of generated sentences may be very similar or even identical, which makes the explanations less personalized to the target users and items. These problems amount to the importance of quality control in natural language generation approaches, since poor explanations may bring negative effects to user acceptance of recommendations and affect the overall experience in recommender systems [111].

In this chapter, we propose a NEural TEmplate (NETE)[1] approach to explainable recommendation, in an attempt to generate both expressive and high-quality explanations by bridging the benefits of both template-based and generation-based approaches. In essence, it is a natural language generation method, and the gener-

[1]Codes available at https://github.com/lileipisces/NETE

ated sentences are template-shaped for quality control, e.g, "the rooftop/harbor is a great place to stay". The explanations can be more targeted and specific, because the generation process is implicitly guided by a "neural template" that are adaptive to the given feature. Table 4.1 shows two example explanations generated by NETE, which are more relevant to the ground-truth among the comparative methods.

Technically, we design a new recurrent neural network architecture named Gated Fusion Recurrent Unit (GFRU), which can incorporate neural templates into the explanation generation process. Specifically, the GFRU in our NETE model consists of three components: two gated recurrent units (GRU) [27] that are responsible for generating respectively the item feature word and the explanation context words (i.e., the template), and a gated fusion unit (GFU) [4] that decides which GRU's word to be emitted at each time step. After the generation process, the context words constitute the *neural template*.

In the following, we first introduce our neural template generation model in Section 4.2 and Section 4.3. Section 4.4 then introduces the experimental setup, while interpretation of the results are provided in Section 4.5. We summarize this chapter in Section 4.6.

## 4.2 Model Description

In this section, we present our proposed NEural TEmplate (NETE) model, which consists of two modules for recommendation and explanation, respectively. An overview of the model is given in Fig. 4.1. The goal of recommendation module is to predict a rating $\hat{r}_{u,i}$, given a user $u$ and an item $i$. Meanwhile, based on a feature $f_{u,i}$ of the item that the user is interested in, our model can generate a template-shaped sentence, which is realized by our proposed Gated Fusion Recurrent Unit (GFRU), as an explanation to the recommendation. The input feature $f_{u,i}$ could be an arbitrary feature that we want the generated explanation to talk about. Depending on the application scenario, it can be either manually set by the user $u$, or predicted by a feature prediction model. In Section 4.3, we provide a

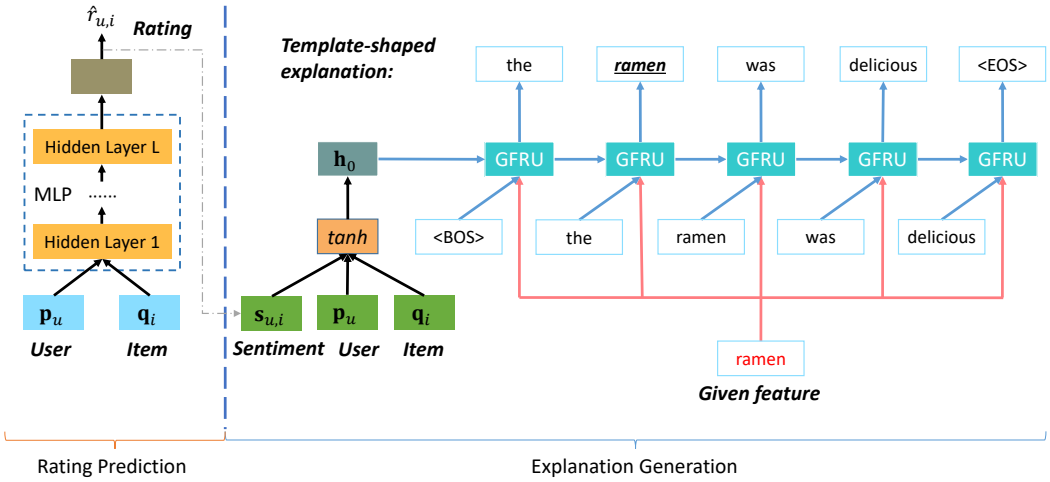



Figure 4.1: Overview of our proposed model NETE that consists of two basic modules for rating prediction (left) and explanation generation (right), respectively. Given a feature, the GFRU component in our model is able to generate a template-shaped explanation that contains the feature.

simple point-wise mutual information (PMI)-based approach for feature prediction. In summary, the training data comprise of users, items, ratings, features and explanation sentences, while during the testing stage, only users, items and features are needed.

Throughout this chapter, scalars are written in italic lower-cases, e.g., $x$, vectors are denoted as bold lower-cases, e.g., $\mathbf{x}$, and matrices are represented as bold upper-cases, e.g., $\mathbf{X}$, no matter whether they carry subscript or superscript or not. In addition, all vectors in this chapter are column vectors, if there is no additional statement. Table 4.2 depicts the key notations and concepts.

### 4.2.1 Personalized Recommendation

Traditionally, rating prediction task is implemented via the inner product between the user and item latent factors [57], but its bi-linear nature may make it difficult to model complex user-item interactions [45]. Therefore, we adopt non-linear transformations that have been shown to have better representation ability in different fields, such as computer vision [58], natural language processing [87] and speech recognition [40]. More specifically, we employ multi-layer perceptron (MLP) with $L$

Table 4.2: Key notations and concepts.

| **Symbol** | **Description** |
|---|---|
| $\mathcal{T}$ | training set |
| $\mathcal{U}$ | set of users |
| $\mathcal{I}$ | set of items |
| $\mathcal{V}$ | set of words |
| $\mathcal{F}$ | set of features |
| $\mathcal{E}$ | set of explanations |
| $\mathbf{p}_u$ | embedding of user $u$ |
| $\mathbf{q}_i$ | embedding of item $i$ |
| $\mathbf{W}$ | weight matrix |
| $\mathbf{w}, \mathbf{b}$ | weight vector |
| $b$ | weight scalar |
| $n$ | dimension of RNN state |
| $d$ | dimension of embedding |
| $L$ | number of hidden layers |
| $r_{u,i}$ | rating assigned by user $u$ on item $i$ |
| $\sigma(\cdot)$ | sigmoid activation function |
| $\tanh(\cdot)$ | hyperbolic tangent function |
| $\text{softmax}(\cdot)$ | softmax function |

hidden layers to capture the interactions between users and items, as shown in the left part of Fig. 4.1. Formally, given the IDs of user $u$ and item $i$, we can obtain their latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ (also called representations or embeddings), and then the recommendation module is defined as:

$$\begin{cases} \mathbf{z}_1 = \sigma(\mathbf{W}_1[\mathbf{p}_u, \mathbf{q}_i] + \mathbf{b}_1) \\ \mathbf{z}_2 = \sigma(\mathbf{W}_2\mathbf{z}_1 + \mathbf{b}_2) \qquad \text{and} \quad \hat{r}_{u,i} = \mathbf{w}_r^\top \mathbf{z}_L + b_r \\ \dots \\ \mathbf{z}_L = \sigma(\mathbf{W}_L\mathbf{z}_{L-1} + \mathbf{b}_L) \end{cases} \tag{4.2.1}$$

where $[\cdot, \cdot]$ denotes the concatenation of vectors, $\sigma(\cdot)$ is the sigmoid activation function, $\mathbf{W}_x \in \mathbb{R}^{2d \times 2d}$ and $\mathbf{b}_x \in \mathbb{R}^{2d}$ are weight matrices and bias vectors in the hidden layers, while $\mathbf{w}_r \in \mathbb{R}^{2d}$ and $b_r \in \mathbb{R}$ correspond to the weight and bias parameters in the final linear layer.

To minimize the difference between ground-truth ratings and the predicted ones, we adopt the mean squared error loss function:

$$\mathcal{L}_r = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} (r_{u,i} - \hat{r}_{u,i})^2 \tag{4.2.2}$$

where $\mathcal{T}$ is the training data set, and $r_{u,i}$ denotes the ground truth rating that user $u$ assigned to item $i$. In this way, the randomly initialized latent vectors $\mathbf{p}_u$ and $\mathbf{q}_i$ can be updated via back-propagation during the training stage.

For personalized recommendation, we can predict ratings for each user's unobserved items, and recommend a number of top-ranked items.

### 4.2.2 Explanation Generation

The module for explanation generation (illustrated in the right part of Fig. 4.1) is compatible with any recommendation module, since it only leverages the predicted ratings resulting from the recommendation module to enforce sentiment control on the generated explanations. In the following, we first introduce the encoder-decoder framework for natural language generation, and then present our proposed Gated Fusion Recurrent Unit (GFRU), which is able to generate template-shaped explanations.

**Encoder-decoder.** The explanation generation problem can be formulated as a table-to-text generation task [121], where the table contains a user, an item, and other attributes, e.g., a rating for the user-item pair. We use user $u$'s representation $\mathbf{p}_u \in \mathbb{R}^d$ and item $i$'s representation $\mathbf{q}_i \in \mathbb{R}^d$ as input to the encoder, so that the decoded word sequence can be personalized to different user-item pairs. Moreover, we also incorporate the predicted rating $\hat{r}_{u,i}$, so as to enforce sentiment control on the generated explanation. Concretely, suppose the rating scale is 1 to 5, following the common practice in recommender systems and sentiment analysis, we map $\hat{r}_{u,i}$ to -1 if the rating is less than 3, otherwise +1. We then represent this sentiment using the corresponding representation $\mathbf{s}_{u,i} \in \mathbb{R}^d$ (there are only two vectors, representing positive and negative sentiment, respectively).

During the training phase, we use the sentiment associated with the given feature in a sentence instead of the overall rating to create the sentiment representation, because we find that the feature sentiment is more consistent with the user's perception on the item than the rating. We will introduce how to obtain the feature sentiments in Section 4.3. To encode the inputs into a vector, we adopt MLP with one hidden layer as the encoder,

$$\mathbf{h}_0 = \tanh(\mathbf{W}_e[\mathbf{p}_u, \mathbf{q}_i, \mathbf{s}_{u,i}] + \mathbf{b}_e) \tag{4.2.3}$$

where $\mathbf{W}_e \in \mathbb{R}^{n \times 3d}$ and $\mathbf{b}_e \in \mathbb{R}^n$ are model parameters, and $\tanh(\cdot)$ denotes the hyperbolic tangent function.

The encoded vector $\mathbf{h}_0$ is used as the initial hidden state of the decoder. Hidden states of the other time steps can be computed by recurrently feeding the representation of the $(t-1)$-th input word $\mathbf{x}_{t-1}$ into the decoder,

$$\mathbf{h}_t = g(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}) \tag{4.2.4}$$

where the hidden vector $\mathbf{h}_{t-1}$ encodes the information of previously generated words, and the decoder $g(\cdot)$ can be recurrent neural networks (RNN), long short-term memory (LSTM) networks [47], or gated recurrent units (GRU) [27]. In this work, we adopt GRU as the decoder, because it shows competitive performance with much better computational efficiency than LSTM [70].

During each step of decoding, the decoder recurrently produces a word based on previously generated words, which can be expressed as,

$$p(y_t|y_{<t}, \mathbf{h}_0) = \text{softmax}_{y_t}(\mathbf{W}_v\mathbf{h}_t + \mathbf{b}_v) \tag{4.2.5}$$

where softmax$(\cdot)$ denotes the softmax function, $\mathbf{W}_v \in \mathbb{R}^{|\mathcal{V}|\times n}$ and $\mathbf{b}_v \in \mathbb{R}^{|\mathcal{V}|}$ are model parameters, $y_{<t}$ represents words produced before time step $t$, and $y_t$ is the word predicted at the current time step. At time step $t$, the decoder takes in the hidden vector $\mathbf{h}_t$ and maps it onto a $|\mathcal{V}|$-sized vector, where $\mathcal{V}$ is the vocabulary of words in the dataset. This vector can be regarded as the probability distribution over the vocabulary, from which a word $y_t$ with the largest probability can be sampled.

**Gated Fusion Recurrent Unit.** Although vanilla decoder can be employed to generate explanations, its generation could be irrelevant to the recommended item, as discussed above. To address this problem, we propose to fuse one feature of the item into the decoding process, which is realized by our proposed Gated Fusion Recurrent Unit (GFRU). Thus, Eq. (4.2.4) can be reformulated as,

$$\mathbf{h}_t = g(\mathbf{x}_{t-1}, \mathbf{h}_{t-1}, \mathbf{x}_f) \tag{4.2.6}$$

where $\mathbf{x}_f$ is the representation of the feature $f \in \mathcal{F}$, and $\mathcal{F} \subset \mathcal{V}$. There are some works [70, 26] that fuse auxiliary information via the initial hidden state, but the long-term dependency problem [8] may make the information's influence become weaker and weaker over time. This problem could be avoided in our model, because the given feature is fed into GFRU at each time step to refresh its memory.

Specifically, our GFRU consists of three components: two GRUs and one gated fusion unit (GFU) [4]. We treat the feature and the neural template as two types of information, so we use two GRUs to process them, which are finally merged by GFU. During explanation generation process, the *context GRU* takes the previously generated word as input, and the *feature GRU* takes the given feature all the time, as shown in Fig. 4.2. The GFU then combines the outputs resulting from the two GRUs to emit a final hidden state, which can be used to predict the next word.

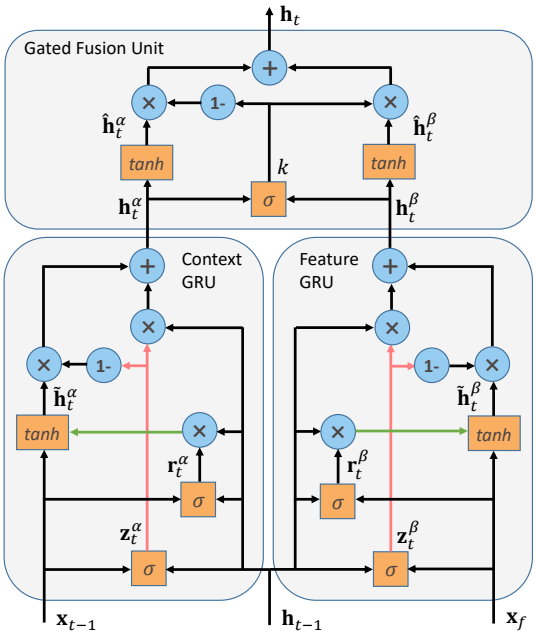



Figure 4.2: The structure of our proposed GFRU decoder with three components. The word produced in the previous step and the given feature are processed by the bottom two GRUs, respectively, whose outputs are merged by the GFU component, which produces a final hidden state for the current time step.

Let $\mathbf{h}_{t-1} \in \mathbb{R}^n$ be the previous hidden state of our GFRU, and $\mathbf{x}_{t-1} \in \mathbb{R}^d$ be the representation of the previously generated word. The hidden state of context GRU at the current time step, $\mathbf{h}_t^\alpha = g^\alpha(\mathbf{x}_{t-1}, \mathbf{h}_{t-1})$, can be computed as follows,

$$\begin{cases} \mathbf{z}_t^\alpha = \sigma(\mathbf{W}_z^\alpha[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}] + \mathbf{b}_z^\alpha) \\ \mathbf{r}_t^\alpha = \sigma(\mathbf{W}_r^\alpha[\mathbf{x}_{t-1}, \mathbf{h}_{t-1}] + \mathbf{b}_r^\alpha) \\ \widetilde{\mathbf{h}}_t^\alpha = \tanh(\mathbf{W}_h^\alpha[\mathbf{x}_{t-1}, \mathbf{r}_t^\alpha \odot \mathbf{h}_{t-1}] + \mathbf{b}_h^\alpha) \\ \mathbf{h}_t^\alpha = \mathbf{z}_t^\alpha \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t^\alpha) \odot \widetilde{\mathbf{h}}_t^\alpha \end{cases} \tag{4.2.7}$$

where $\mathbf{W}_x^\alpha \in \mathbb{R}^{n \times (d+n)}$ and $\mathbf{b}_x^\alpha \in \mathbb{R}^n$ are model parameters, $\mathbf{z}_t^\alpha$ and $\mathbf{r}_t^\alpha$ control how much of the past information to keep and forget, respectively, and $\odot$ denotes element-wise multiplication. Accordingly, with the previous hidden state of GFRU $\mathbf{h}_{t-1} \in \mathbb{R}^n$ and the representation of the feature $\mathbf{x}_f \in \mathbb{R}^d$, the current hidden state of the feature GRU is defined as $\mathbf{h}_t^\beta = g^\beta(\mathbf{x}_f, \mathbf{h}_{t-1})$. Notice that, the two GRUs do not share parameters, so we use superscripts $\alpha$ and $\beta$ to differentiate them.

Then, we integrate the two types of decoding information, i.e., $\mathbf{h}_t^\alpha$ and $\mathbf{h}_t^\beta$, into

the final hidden state $\mathbf{h}_t$ via the GFU. The computing equations are as follows,

$$\begin{cases} \hat{\mathbf{h}}_t^{\alpha} = \tanh(\mathbf{W}_{\alpha}\mathbf{h}_t^{\alpha}) \\ \hat{\mathbf{h}}_t^{\beta} = \tanh(\mathbf{W}_{\beta}\mathbf{h}_t^{\beta}) \\ k = \sigma(\mathbf{w}_k^{\top}[\hat{\mathbf{h}}_t^{\alpha}, \hat{\mathbf{h}}_t^{\beta}]) \\ \mathbf{h}_t = (1-k) \odot \mathbf{h}_t^{\alpha} + k \odot \mathbf{h}_t^{\beta} \end{cases} \tag{4.2.8}$$

where $\mathbf{W}_{\alpha} \in \mathbb{R}^{n \times n}$, $\mathbf{W}_{\beta} \in \mathbb{R}^{n \times n}$ and $\mathbf{w}_k \in \mathbb{R}^{2n}$ are parameters to be learned. From Eq. (4.2.8), we can see that the weight $k$ automatically controls the decoding information of the two GRUs. When $k$ is small, the output of GFRU mainly comes from the context GRU to produce a template-shaped word sequence. Conversely, when it is large, our GFRU relies on feature GRU to fill the given feature in the template. With the GFRU that is able to perform generation according to an item feature, we can improve the quality of the generated explanation by making it more relevant to the recommended item.

**Loss Function.** To train the module of explanation generation, we draw on the widely used Negative Log-Likelihood (NLL) loss function, and compute the loss for each user-item pair in the training set,

$$\mathcal{L}_e = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} \frac{1}{|E_{u,i}|} \sum_{t=1}^{|E_{u,i}|} -\log p(y_t) \tag{4.2.9}$$

where $E_{u,i}$ is the ground-truth explanation for the user-item pair ($u$, $i$), $|E_{u,i}|$ is its length in number of words, and $p(y_t)$ denotes the predicted probability of word $y_t$ from Eq. (4.2.5).

### 4.2.3 Model Training

In general, our explainable recommendation framework in Fig. 4.1 involves two modules for two tasks – the recommendation task and the explanation task. As discussed in Section 4.2.2, our explanation module is compatible with any rating prediction model. To learn the model parameters, one option is to jointly train the two tasks in the manner of multi-task learning. But to make our approach general

enough so as to study the compatibility with different recommendation models, we propose to train them separately in a sequential manner. Specifically, we first optimize the loss function of the recommendation task (Eq. (4.2.2)) based on the user-item pairs in the training data, followed by that of the explanation task (Eq. (4.2.9)). Notice that, since the two tasks are separated, the representations of users and items (i.e., $\mathbf{p}_u$ for user $u$ and $\mathbf{q}_i$ for item $i$) in the two tasks are different sets of latent vectors. During the testing stage, the predicted ratings resulting from the recommendation module are used as the input sentiment for the explanation task.

## 4.3 Feature Prediction based on PMI

As we discussed in the previous section, our NETE model can generate an explanation that talks about a given feature. According to different application scenarios, this feature could be either manually specified, or predicted from data. In this section, we provide a simple method to predict the feature for the target item, on which the user did not have interaction. It particularly considers the relationship between features in the user's historical reviews and those in the item's reviews.

We first apply a sentiment analysis toolkit[2] [134] to extract features (i.e., aspects) and their associated sentiments from user reviews, e.g., (rooms, spacious, +1) from the sentence "The rooms are spacious", where "rooms" is a feature word, "spacious" is an opinion word, and +1 means that the feature-opinion pair expresses a positive sentiment. We denote the collection of all extracted features as $\mathcal{F}$. As point-wise mutual information (PMI) has been widely used in computational linguistics to find the association between words/features [89, 128], we employ this approach to predict a user's interests in a feature by comparing it with features mentioned in the user's reviews.

Formally, given two features $f_u$ and $f_i$ associated with user $u$ and item $i$, respec-

[2]Our procedure to create (feature, opinion, sentence, sentiment) quadruples from user reviews and the already created datasets are available at https://github.com/lileipisces/Sentires-Guide.

tively, the PMI is computed as:

$$\text{PMI}(f_u, f_i) = \log \frac{p(f_u, f_i)}{p(f_u)p(f_i)} = \log \frac{p(f_u|f_i)}{p(f_u)} \tag{4.3.10}$$

Then, we select the feature $\hat{f}_i$ with the largest PMI score from item $i$'s feature collection $\mathcal{F}_i$ as prediction, which is achieved by comparing against user $u$'s features $\mathcal{F}_u$, i.e., $\hat{f}_i = \text{argmax}_{f \in \mathcal{F}_i} \text{PMI}(\mathcal{F}_u, f)$, where

$$\begin{aligned} \text{PMI}(\mathcal{F}_u, f) &= \log \frac{p(\mathcal{F}_u|f)}{p(\mathcal{F}_u)} \approx \log \frac{\prod_{f' \in \mathcal{F}_u} p(f'|f)}{\prod_{f' \in \mathcal{F}_u} p(f')} \\ &= \sum_{f' \in \mathcal{F}_u} \log \frac{p(f'|f)}{p(f')} = \sum_{f' \in \mathcal{F}_u} \text{PMI}(f', f) \end{aligned} \tag{4.3.11}$$

The approximation in Eq. (4.3.11) is based on the independence assumption between the prior distribution $p(f')$ and posterior distribution $p(f'|f)$. The assumption may not always hold, but we use them in a pragmatic way, so that feature-level PMI on user $u$'s features is additive. In particular, feature-level PMI penalizes a frequently occurring feature by dividing its prior probability, which could help us filter out less informative features for producing better explanations. In implementation, we use a feature's occurring frequency in all the user reviews to compute the feature-level PMI:

$$\text{PMI}(f', f) = \frac{\text{Freq}(f', f)}{\text{Freq}(f') \cdot \text{Freq}(f)} \tag{4.3.12}$$

where $\text{Freq}(f', f)$ denotes two features' co-occurring frequency.

## 4.4 Experimental Setup

### 4.4.1 Datasets

In our experiments, we use three real-world datasets from different domains, i.e., hotel, restaurant and movie, to evaluate our proposed model. For the hotel domain, we construct the dataset with reviews crawled from a travel website TripAdvisor[3]. Specifically, we implement a crawler to collect all the user reviews associated with

[3]https://www.tripadvisor.com

Table 4.3: Statistics of the datasets.

| | **TripAdvisor** | **Yelp** | **Amazon** |
|---|---|---|---|
| # of users | 9,765 | 27,147 | 7,506 |
| # of items | 6,280 | 20,266 | 7,360 |
| # of reviews | 320,023 | 1,293,247 | 441,783 |
| # of features | 5,069 | 7,340 | 5,399 |
| Avg. # of reviews / user | 32.77 | 47.64 | 58.86 |
| Avg. # of reviews / item | 50.96 | 63.81 | 60.02 |
| Avg. # of words / explanation | 13.01 | 12.32 | 14.14 |

every hotel located in an international city Hong Kong. To obtain past interactions of the users appeared in these reviews, the crawler subsequently collects their historical reviews. We only keep English reviews, which gives us 2,118,108 records in total. For restaurant domain, we use the dataset from Yelp Challenge 2019[4]. This publicly available dataset consists of 6,685,900 restaurant reviews written by 1,637,138 users for 192,606 businesses located in 10 metropolitan areas. The last dataset for the movie domain is from Amazon 5-core[5] Movies & TV, which contains 1,697,533 reviews by 123,960 users for 50,052 items.

Since the three datasets are very large, we further process them by recursively removing users and items with less than 20 interactions, which results in three subsets. Each review record in our datasets comprises of user ID, item ID, overall rating in the scale of 1 to 5, and textual review. After extracting features from user reviews, for each record we select one sentence containing at least one feature from the review as the ground-truth explanation. During the testing stage, we assume that the features in the explanations are manually set by the users themselves. The key characteristics of the three datasets after processing are presented in Table 4.3.

[4]https://www.yelp.com/dataset/challenge

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

### 4.4.2 Evaluation Metrics

To measure the recommendation accuracy of different methods, we adopt two widely used metrics: Root Mean Square Error (**RMSE**) and Mean Absolute Error (**MAE**). For both metrics, a lower value indicates a better performance.

As to the explainability, we evaluate the generated explanations from two perspectives: the relevance to ground-truth text and the degree of personalization. For the first perspective, we adopt two commonly used metrics, i.e., **BLEU** [91] in machine translation and **ROUGE** [75] in text summarization, to estimate the overlapping of text segments between a generated explanation and the ground-truth. For the evaluation in different granularities, we report the results of BLEU-1 and BLEU-4, and use Precision, Recall and F1 of ROUGE-1 and ROUGE-2. The larger BLEU and ROUGE scores are, the closer the generated explanations are to the ground-truth.

For the second perspective, i.e., personalization degree, we propose four metrics: the ratio of unique sentences, the ratio of explanations that contain the given features, the ratio of distinct features in all the generated explanations, and the diversity of features between any two explanations:

1. **Unique Sentence Ratio (USR)**. As discussed before, we find that the generated sentences in existing methods tend to be similar, i.e., many user-item pairs have exactly the same explanation. To examine how severe the problem is, we present this metric that calculates how many unique sentences are generated.

$$USR = |\mathcal{E}| / N \tag{4.4.13}$$

where $\mathcal{E}$ denotes the set of unique explanations generated by a model, and $N$ is the total number of testing samples. Notice that, only the exactly matched sentences are considered being identical, and as a result only one of them is added to $\mathcal{E}$.

2. **Feature Matching Ratio (FMR)**. Besides sentence-level evaluation, we also evaluate the generated explanations at feature-level. Since a feature is given for each user-item pair as input to our model, we are interested in whether it is really

included in the generated explanation, which can be formulated as follows:

$$FMR = \frac{1}{N} \sum_{u,i} \delta(f_{u,i} \in \hat{E}_{u,i}) \tag{4.4.14}$$

where $\hat{E}_{u,i}$ is the generated explanation for the user-item pair, $f_{u,i}$ is the given feature, and $\delta(x) = 1$ if $x$ is true and $\delta(x) = 0$ otherwise.

3. **Feature Coverage Ratio (FCR)**. We propose this metric to measure the features at corpus-level. Intuitively, the more features in the ground-truth explanations a model generates, the better preference modeling capability the model has. Therefore, this metric estimates the ratio of features in the produced explanations, as compared with those in a whole dataset:

$$FCR = N_g / |\mathcal{F}| \tag{4.4.15}$$

where $\mathcal{F}$ is the collection of unique features in ground-truth explanations, and $N_g$ is the number of distinct features appeared in the generated explanations.

4. **Feature Diversity (DIV)**. It is reasonable that explanations generated for different user-item pairs do not always discuss about the same feature. To estimate how diverse features contained in explanations are, we present this metric. Let $\hat{\mathcal{F}}_{u,i}$ and $\hat{\mathcal{F}}_{u',i'}$ respectively denote two sets of features in two generated explanations, we can measure their difference. To calculate the feature diversity in the testing data, we estimate the difference between any two feature sets as follows,

$$DIV = \frac{2}{N \times (N-1)} \sum_{u,u',i,i'} \left| \hat{\mathcal{F}}_{u,i} \cap \hat{\mathcal{F}}_{u',i'} \right| \tag{4.4.16}$$

where $\cap$ represents the intersection of two sets, and $|\cdot|$ denotes the number of elements in the resulting set.

A lower DIV indicates a smaller overlap between feature sets, and thus a higher diversity. For the other metrics, i.e., USR, FMR and FCR, the higher the scores are, the better the performance is. Overall, the four proposed metrics evaluate explanations from different perspectives, and all of them can be applied to other explanation generation methods.

### 4.4.3 Compared Methods

Since our key focus in this work is explanation generation, we first introduce comparative methods in this regard. Specifically, we compare our model with two state-of-the-art natural language generation models, i.e., Neural Rating and Tips generation (NRT) [70] and Attribute-to-sequence (Att2Seq) [31]. We omit other natural language generation models, whose inputs are different from ours, which makes them not directly comparable. For example, Visually Explainable Collaborative Filtering (VECF) [22] and Multimodal Review Generation (MRG) [112] generate text based on image features, while in [120] neighborhood relation between reviews is required for review generation. We also provide two variants of our own model for ablation study.

- **NRT**: Neural Rating and Tips generation [70]. This model adopts multi-layer perceptron (MLP) to predict a rating for a user-item pair, and formulates the explanation generation problem as a text (i.e., tip) summarization task. The two tasks of recommendation and tip generation are integrated into a multi-task learning framework. In our implementation, the explanation sentence is regarded as tip.

- **Att2Seq**: Attribute-to-Sequence [31] employs MLP to encode three attributes, i.e., user, item and rating, and adopts two-layer LSTM to decode the encoded representations for generating a textual review. In our implementation, the LSTM is replaced by GRU for the consistency with our model, and the explanation sentence is treated as review. We also disable its attention mechanism, because it makes the generated text not quite readable in our experimental trials.

- **NETE-GRU**: This is a variant of our model NETE, where the decoder is a standard GRU instead of our proposed GFRU. By comparing with this variant, we could know whether GFRU truly helps with explanation generation.

- **NETE-PMI**: The only difference between this variant and the standard NETE

model is that its input features are predicted by the PMI method introduced in Section 4.3, while NETE uses the feature given by a ground-truth explanation to see whether it is able to generate a similar sentence commenting about this feature.

To evaluate the recommendation performance, in addition to NRT (NETE-GRU and NETE-PMI are excluded because their recommendation module is the same as NETE's), we compare with the following four typical rating prediction methods:

- **PMF**: Probabilistic Matrix Factorization [88]. This is the standard matrix factorization method that characterizes users and items by latent factors inferred from observed ratings. We use alternative least square (ALS) to optimize its objective function.

- **SVD++**: Singular Value Decomposition [56]. It extends matrix factorization by regarding items that a user interacted with as implicit feedback, and integrates them into the latent factor modeling.

- **ConvMF**: Convolutional Matrix Factorization [52]. It employs a convolutional neural network (CNN) to exploit textual information from item description for enhancing PMF. We concatenate user reviews of an item as its description.

- **DeepCoNN**: Deep Cooperative Neural Networks [136]. This method learns the representations of users and items from their aggregated reviews via two parallel convolutional neural networks (CNN) [54].

### 4.4.4 Implementation Details

We randomly split each dataset into training (80%), validation (10%) and testing (10%) sets, and ensure that there is at least one instance in the training set for each user/item. We repeat the splitting process for 5 times, and report the average performance on the testing set, while the validation set is used for hyper-parameters

tuning. The early stopping strategy is performed for all the models, i.e., a model's results on the testing set are reported, when its loss on the validation set reaches the minimum.

We implement all the methods in Python. All the neural methods, i.e., ConvMF, DeepCoNN, Att2Seq, NRT and NETE, are implemented using TensorFlow, and optimized by Adam [55] with $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. For MF-based models, i.e., PMF, SVD++ and ConvMF, we set the number of latent factors to 20 (optimal choice on our datasets). The regularization parameter of PMF and ConvMF are searched from [0.1, 1, 10, 100], while both the regularization parameter and learning rate of SVD++ are searched from [0.1, 0.01, 0.001]. For review-based methods, i.e., ConvMF and DeepCoNN, we set the maximum document length of aggregated reviews to 1,000 words. For all the models that make use of text, we select top 20,000 distinct words with the largest frequency on the training set to construct the vocabulary $\mathcal{V}$. For all the natural language generation models, i.e., Att2Seq, NRT and NETE, we set the maximum length of generated text to 15, which is reasonable as the average length of explanation sentences is around 13 (as shown in Table 4.3). Another reason of limiting the sentence length is that presenting too much information may overwhelm users [119, 46]. We reuse the other hyper-parameters of the baselines as reported in the original papers.

For our model NETE, the learning rate is set to 0.0001 and the batch size 128. We set $d$ (the dimension of user/item/sentiment/word representations) as 200, $n$ (the dimension of RNN hidden states) as 256, and $L$ (the number of MLP layers for rating prediction) as 4. For regularization, the dropout ratio of RNN is set to 0.2.

## 4.5 Results and Analysis

In this section, we first present quantitative evaluation on the generated explanations (evaluated via both automatic metrics and human evaluators), followed by a qualitative analysis on explanation case studies. At last, we evaluate the recommendation performance, as well as the feature prediction performance.

Table 4.4: Performance comparison of all natural language generation methods in terms of Personalization, BLEU (%) and ROUGE (%) on TripAdvisor dataset. For feature diversity (DIV), a lower value indicates a better performance, while for the other metrics, the larger, the better. The best performing values are boldfaced. Improvements are made by NETE over the best baseline (** indicates the statistical significance for $p < 0.01$ via Student's $t$-test). Note that BLEU, ROUGE and improvement scores in the table are percentage values (i.e., 14.2 means 14.2%), while USR, FMR, FCR and DIV scores are absolute values (i.e., 0.18 means 0.18).

| | Personalization | | | | BLEU (%) | | ROUGE-1 (%) | | | ROUGE-2 (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | USR | FMR | FCR | DIV | BLEU-1 | BLEU-4 | Precision | Recall | F1 | Precision | Recall | F1 |
| NRT | 0.00 | - | 0.00 | 13.61 | 14.26 | 0.80 | 17.57 | 16.52 | 16.56 | 2.45 | 2.64 | 2.48 |
| Att2Seq | 0.18 | - | 0.17 | 3.93 | 14.76 | 1.01 | 19.26 | 14.45 | 15.83 | 2.43 | 1.96 | 2.06 |
| NETE-GRU | 0.27 | - | 0.15 | 3.00 | 13.84 | 0.92 | 18.55 | 13.64 | 15.02 | 2.23 | 1.76 | 1.86 |
| NETE-PMI | **0.79** | 0.38 | **0.30** | 2.92 | 14.55 | 0.82 | 17.84 | 13.96 | 14.90 | 2.01 | 1.70 | 1.74 |
| **NETE** | 0.57** | **0.78** | 0.27** | **2.22**** | **22.39**** | **3.66**** | **35.68**** | **24.86**** | **27.71**** | **10.20**** | **6.98**** | **7.66**** |
| Improvement (%) | +210.7 | - | +57.1 | +77.1 | +51.7 | +261.3 | +85.2 | +50.5 | +67.3 | +317.0 | +164.0 | +209.1 |

Table 4.5: Performance comparison of all natural language generation methods in terms of Personalization, BLEU (%) and ROUGE (%) on Yelp dataset. For feature diversity (DIV), a lower value indicates a better performance, while for the other metrics, the larger, the better. The best performing values are boldfaced. Improvements are made by NETE over the best baseline (** indicates the statistical significance for $p < 0.01$ via Student's $t$-test). Note that BLEU, ROUGE and improvement scores in the table are percentage values (i.e., 14.2 means 14.2%), while USR, FMR, FCR and DIV scores are absolute values (i.e., 0.18 means 0.18).

| | Personalization | | | | BLEU (%) | | ROUGE-1 (%) | | | ROUGE-2 (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | USR | FMR | FCR | DIV | BLEU-1 | BLEU-4 | Precision | Recall | F1 | Precision | Recall | F1 |
| NRT | 0.00 | - | 0.00 | 3.72 | 3.96 | 0.19 | 19.25 | 8.03 | 10.95 | 1.30 | 0.51 | 0.69 |
| Att2Seq | 0.14 | - | 0.12 | 2.19 | 10.41 | 0.59 | 18.20 | 11.38 | 13.21 | 1.81 | 1.16 | 1.31 |
| NETE-GRU | 0.28 | - | 0.14 | 1.98 | 11.12 | 0.64 | 16.85 | 11.60 | 13.00 | 1.68 | 1.19 | 1.30 |
| NETE-PMI | **0.64** | 0.58 | **0.28** | 1.65 | 10.62 | 0.56 | 15.37 | 10.80 | 12.01 | 1.49 | 1.05 | 1.15 |
| **NETE** | 0.52** | **0.80** | 0.27** | **1.48**** | **19.31**** | **2.69**** | **33.98**** | **22.51**** | **25.56**** | **8.93**** | **5.54**** | **6.33**** |
| Improvement (%) | +257.6 | - | +116.3 | +48.0 | +85.5 | +355.3 | +76.5 | +97.8 | +93.4 | +393.1 | +377.6 | +384.0 |

Table 4.6: Performance comparison of all natural language generation methods in terms of Personalization, BLEU (%) and ROUGE (%) on Amazon dataset. For feature diversity (DIV), a lower value indicates a better performance, while for the other metrics, the larger, the better. The best performing values are boldfaced. Improvements are made by NETE over the best baseline (** and * respectively indicate the statistical significance for $p < 0.01$ and $p < 0.05$ via Student's $t$-test). Note that BLEU, ROUGE and improvement scores in the table are percentage values (i.e., 14.2 means 14.2%), while USR, FMR, FCR and DIV scores are absolute values (i.e., 0.18 means 0.18).

| | Personalization | | | | BLEU (%) | | ROUGE-1 (%) | | | ROUGE-2 (%) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | USR | FMR | FCR | DIV | BLEU-1 | BLEU-4 | Precision | Recall | F1 | Precision | Recall | F1 |
| NRT | 0.00 | - | 0.01 | 5.46 | 14.02 | 0.57 | 23.57 | 14.24 | 16.87 | 2.53 | 1.70 | 1.92 |
| Att2Seq | 0.34 | - | 0.18 | 2.81 | 12.78 | 1.01 | 20.53 | 13.49 | 15.42 | 2.77 | 1.87 | 2.09 |
| NETE-GRU | 0.38 | - | 0.11 | 2.34 | 12.10 | 0.95 | 20.16 | 12.93 | 14.93 | 2.63 | 1.75 | 1.97 |
| NETE-PMI | **0.72** | 0.50 | **0.19** | 3.06 | 13.02 | 0.82 | 20.93 | 12.76 | 14.99 | 2.36 | 1.63 | 1.81 |
| **NETE** | 0.57** | **0.71** | **0.19*** | **1.93**** | **18.76**** | **2.46**** | **33.87**** | **21.43**** | **24.81**** | **7.58**** | **4.77**** | **5.46**** |
| Improvement (%) | +69.1 | - | +5.6 | +45.2 | +33.8 | +143.6 | +43.7 | +50.5 | +47.1 | +174.3 | +154.9 | +161.2 |

### 4.5.1 Quantitative Analysis on Explanations

Evaluation results of the explanations as generated by all the natural language generation methods on the three datasets are shown in Tables 4.4, 4.5 and 4.6. We first analyze the personalization degree in terms of the four metrics, i.e., USR, FMR, FCR, and DIV.

As it can be seen, our NETE model and its variant NETE-PMI generally perform better than the other methods on different metrics, especially on USR, which measures the ratio of generated unique sentences. By comparing the two baselines NRT and Att2Seq, we can find that the former generates less than 1% unique sentences on each dataset, while the latter produces diverse sentences with much higher USR. In the meantime, NRT's performance on ROUGE and BLEU is generally as good as Att2Seq, which evidences that ROUGE and BLEU could not properly evaluate sentence diversity. Because of our proposed GFRU component, our NETE model generates approximately 55% unique sentences on the three datasets, which shows the capability of our model in generating diverse explanations.

On the other three metrics, all the models show a similar trend as on USR. Again, owing to the GFRU design in our model, the feature coverage ratio (FCR) and feature diversity (DIV) of explanations are largely improved. Moreover, in terms of feature matching ratio (FMR), approximately 75% of the explanations generated by our model contain the given features, which implies its good controllability to comment about these features.

Notably, we observe that NETE-PMI performs better than NETE on USR and FCR. The input features to NETE-PMI are predicted, and they may not exactly match those in the testing samples. Hence, the user-item-feature combination may not be commonly seen in the training data, and the generated explanations and their contained features could be more diverse, resulting in higher USR and FCR. With respect to the other two metrics, i.e., FMR and DIV, NETE-PMI's performance is also competitive to baselines. This variant shows our model's capability in dealing with unseen features, which is quite common in real-world scenarios.

Finally, we analyze the results on BLEU and ROUGE. As we can see, our NETE model consistently outperforms all the baselines/variants on three datasets. We attribute this to the effectiveness of our GFRU module in generating template-shaped explanations that are more relevant to the ground-truth. Generally, NRT, Att2Seq and NETE-GRU achieve the same performance on three datasets, because they all adopt GRU for natural language generation. Our model improves their performance by a large margin, notably with over 100% improvements regarding BLUE-4 and ROUGE-2, which focus on the overlapping of $n$-grams between generated text and the ground-truth. This shows that our model is able to produce high-quality explanations that are semantically much closer to the ground-truth, when the given features are consistent with those in the testing samples. On the other hand, NETE-PMI that takes as input the features predicted by the PMI method, reaches the similar performance as the GRU-based methods, because these features may not always match those in the testing samples (e.g., "room" vs. "location"), and thus the generated explanations may talk about other topics different from those in the ground-truth sentences. If we have an ideal feature predictor that can accurately predict the features in the ground-truth, then our model can generate better explanations than the baselines. But since feature prediction is not the key focus of this work, we only provide a simple PMI which unfortunately could not always perfectly predict the features in the testing samples.

### 4.5.2 Human Evaluation on Explanations

To investigate whether the generated explanations are truly helpful to users in the context of recommendation, we conduct a small-scale user survey on Yelp dataset, since it is about food that people consume every day.

Specifically, we prepare two questions, each containing 20 cases that are randomly sampled from the testing set. We invite 10 volunteers (most are postgraduate students in our department) to evaluate the results. The first question (Q1) is a pair-wise evaluation task, where for each case we ask the participants to choose one

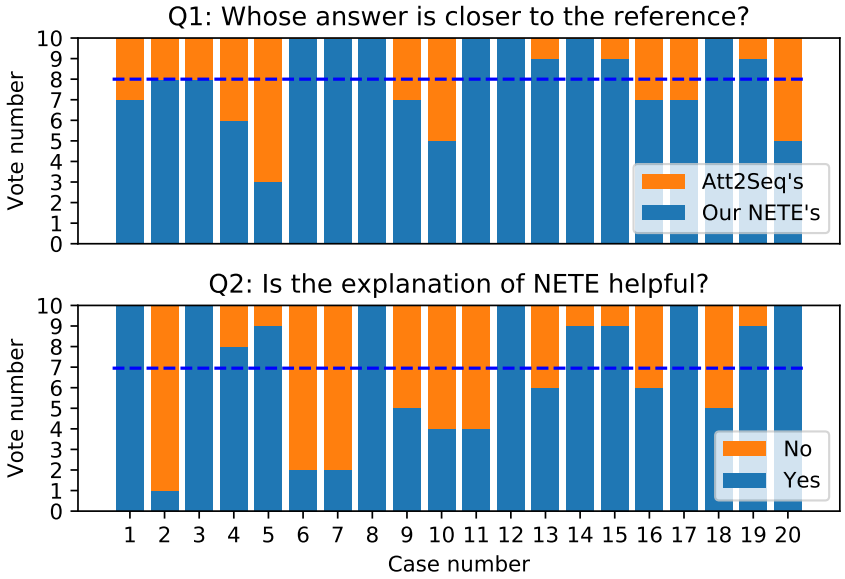



Figure 4.3: Results of human evaluation on generated explanations on Yelp dataset. The blue dotted lines show the average votes for the blue option across 20 cases.

from two explanations respectively generated by our model NETE and the baseline Att2Seq[6] [31] in terms of the explanation's similarity to the given reference. Notice that, to make the comparison fair, we hide the details of models and inform the participants that the answers are randomly shuffled. This task is to investigate whether our model could generate high-quality explanations relative to the baseline. After that, a point-wise evaluation (Q2) on the explanations generated by our model NETE is performed, for which we ask the participants to judge whether the explanation is helpful for them to evaluate the feature of a recommended restaurant. The two questions are listed below:

- **Q1**: Which answer is closer to the reference sentence in terms of semantic similarity?

- **Q2**: Assume you are interested in one feature of a restaurant, do you think the generated explanation is helpful for you to evaluate that feature?

[6]We omit NRT for comparison because its explanations are identical, which may not bring positive results when judged by human evaluators.

We depict the results in Fig. 4.3. The bar chart for Q1 shows that our model obtains 8 votes on average across those 20 cases, which is obviously higher than 2 votes obtained by Att2Seq, which may be because our model is able to integrate the given features into the generation process, making the explanations more relevant to the reference sentences. For Q2, the votes are different from case to case, but on average 7 participants agree that our model explains the given features clearly. This verifies that to a large extent our model is competent to generate useful explanations that can help users better understand the recommended products.

### 4.5.3 Qualitative Case Study on Explanations

In this subsection, we present four groups of generated explanations on the TripAdvisor dataset in Table 4.7 to show our model's good controllability in terms of controlling the explanations to talk about certain features. Results on the other two datasets show similar pattern.

As we can see from the first group, when our model is given different features, i.e., "bathroom", "tub" and "rooms", the generated explanations are not only different but also highly relevant to the features, which shows that our model can generate more targeted explanations for the given features. By comparing the last generated sample in the first group with that in the second group (both about feature "rooms" but the user-item pairs are different), the model generates explanations that describe the same feature in different expressions, which shows that our model is able to produce explanations personalized for different user-item pairs. The last two groups with predicted ratings lower than 3, i.e., negative sentiment, show that our model is capable of controlling the sentiment of the generated explanations. Admittedly, our model is not flawless, because in the last group the predicted rating deviates too much from the ground-truth (2.76 v.s. 4). We provide in-depth analysis on recommendation performance in the next subsection. Overall, these observations manifest our model's controllability in terms of generating explanations corresponding to the given user, item, feature and sentiment.

Table 4.7: Example explanations generated by our NETE model on the TripAdvisor dataset. The first line of each group shows the ground-truth rating and explanation, while other lines show the predicted ratings, the given features, and the generated explanations, where rating $< 3$ denotes negative sentiment and $\geq 3$ positive sentiment. We highlight the mentioned feature in the generated explanations.

| Rating | Feature | Explanation |
|---|---|---|
| **4** | | **The rooms are spacious and the bathroom has a large tub.** |
| 3.90 | bathroom | The **bathroom** was large and had a separate shower. |
| | tub | The bathroom had a separate shower and **tub**. |
| | rooms | The **rooms** are large and comfortable. |
| **4** | | **The rooms are brilliant and ideal for business travellers.** |
| 4.13 | rooms | The **rooms** are very spacious and the **rooms** are very comfortable. |
| **2** | | **The broken furniture and dirty surfaces are a dead giveaway.** |
| 2.96 | furniture | The **furniture** is worn. |
| **4** | | **Ideal for plane spotters and very close to the airport.** |
| 2.76 | airport | It is not close to the **airport**. |

Moreover, there exist common expressions in the generated sentences, e.g., "__ was large/comfortable", which constitute templates as learned from data rather than manually defined. This shows that our model can generate template-shaped explanations that automatically adapt to the input features. Overall, the linguistic quality of the explanations is satisfactory.

Table 4.8: Performance comparison of all the rating prediction methods in terms of RMSE and MAE. The best performing values are boldfaced.

| | TripAdvisor | | Yelp | | Amazon | |
|---|---|---|---|---|---|---|
| | RMSE | MAE | RMSE | MAE | RMSE | MAE |
| | **Traditional methods** | | | | | |
| PMF | 0.870 | 0.696 | 1.086 | 0.877 | 1.034 | 0.807 |
| SVD++ | 0.798 | 0.610 | 1.011 | 0.785 | 0.965 | **0.718** |
| | **Neural methods** | | | | | |
| ConvMF | 0.799 | 0.613 | 1.024 | 0.807 | 0.978 | 0.751 |
| DeepCoNN | 0.796 | 0.607 | 1.011 | 0.789 | 0.959 | 0.721 |
| NRT | **0.792** | **0.605** | **1.007** | **0.783** | **0.957** | **0.718** |
| | **Ours** | | | | | |
| **NETE** | **0.792** | 0.608 | 1.010 | 0.789 | 0.961 | 0.727 |

### 4.5.4 Recommendation Performance

The recommendation performance of our model as compared with baselines is shown in Table 4.8. The observations on the three datasets are consistent. In particular, NRT generally achieves the best performance, because it trains the two tasks of rating prediction and explanation generation in a multi-task learning framework, where the recommendation task benefits the additional textual information from the explanation task via the shared user and item representations. In comparison, NETE trains the two tasks separately, so it forbids the rating prediction task to access such information, but its recommendation accuracy is still comparable to NRT's. Therefore, we believe the sacrificed minor accuracy is tolerable.

We also see that deep neural methods (i.e., NETE, NRT, ConvMF and DeepCoNN) generally perform better than traditional shallow methods (PMF and SVD++), because the non-linear transformations in the former have better representation learning ability [45]. In addition, we notice that the performance gap between tradi-

Table 4.9: Performance comparison of two feature prediction methods in terms of Precision (Pre) and Recall (Rec). The best performing values are boldfaced.

| | TripAdvisor | | Yelp | | Amazon | |
|---|---|---|---|---|---|---|
| | Pre@10 | Rec@10 | Pre@10 | Rec@10 | Pre@10 | Rec@10 |
| RAND | 0.050 | 0.108 | 0.031 | 0.097 | 0.040 | 0.120 |
| **PMI** | **0.122** | **0.256** | **0.112** | **0.342** | **0.081** | **0.246** |

tional methods and review-based methods (ConvMF and DeepCoNN) is small, which may be because review-based methods are mainly to address sparsity problem, but the three datasets are made relatively dense for studying explanation generation. Among traditional methods, SVD++ is better than PMF since the former takes item IDs as implicit feedback into latent factor modeling. Regarding review-based methods, DeepCoNN outperforms ConvMF because the former considers reviews for both user and item modeling, while the latter only models reviews of the item which may be insufficient.

### 4.5.5 Feature Prediction Analysis

To demonstrate the effectiveness of our feature prediction method (denoted as **PMI**) proposed in Section 4.3, we compare the predicted features with those appeared in the corresponding reviews. Specifically, with feature scores computed via Eq. (4.3.11), for each user-item pair we can obtain a ranked feature list. Then, we keep the top 10 as predictions. For comparison, we design a baseline method named **RAND** that randomly selects 10 features from the target item's feature set for each testing user-item pair. To evaluate the two methods, we adopt Precision and Recall.

The results on the three datasets are shown in Table 4.9. Our PMI method consistently performs two times better than RAND. This is as expected, because our method takes both users' and items' features into consideration when computing the PMI values, while RAND arbitrarily selects features from the target item's feature

set without considering the users, which would result in non-personalized feature lists and therefore low scores on Precision and Recall.

## 4.6 Summary

In this chapter, we aim to improve both the expressiveness and the quality of recommendation explanations. To this end, we propose NETE – a NEural TEmplate explanation generation framework that bridges the benefits of both template-based approaches and natural language generation approaches. We not only evaluate the generated explanations based on traditional text quality measures such as BLEU and ROUGE, but also on innovative metrics that evaluate the sentence uniqueness, feature matching, feature coverage, and feature diversity. Experimental results show that our approach is highly controllable to generating explanations that talk about the given user, item, sentiment, and feature.

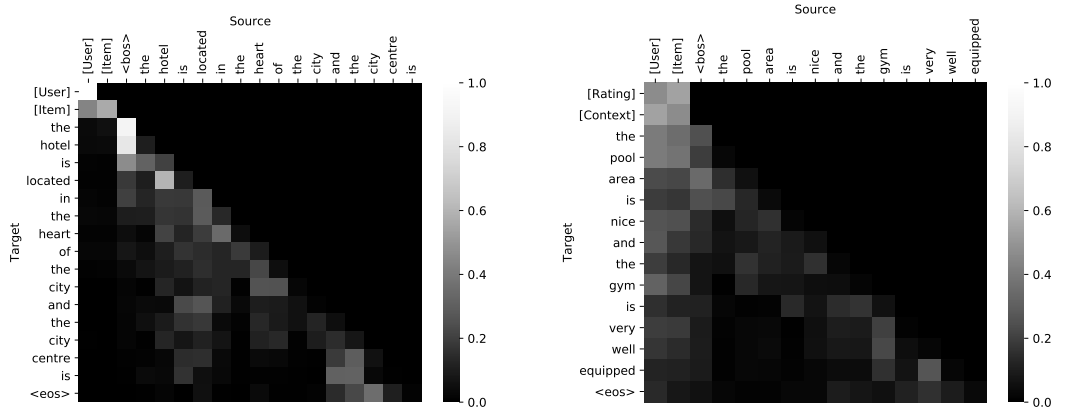
# Chapter 5

# Natural Language Explanation Generation

## 5.1 Background

In the previous chapter, we introduced a neural template explanation generation approach. In order to generate template-like explanations, this approach requires item features specified in advance, which may not always be available. To address this problem, in this chapter we design a more general natural language generation approach, where items features are optional, while user and item IDs are mandatory, since they are crucial identifiers for distinguishing one user/item from the others. Specifically, different users may care about different item features (e.g., "style" vs. "quality"), and different items may have different characteristics (e.g., "fashionable" vs. "comfortable"). An example of explanations for a given pair of user ID and item ID could be "the style of the jacket is fashionable".

We implement this idea with Transformer [114], given its strong language modeling ability as demonstrated on a variety of tasks [94, 30, 10]. However, it would be problematic to directly put IDs and words in the target explanation together for attention learning, since they are in very different semantic spaces. In doing so, the IDs are treated as words, but the IDs appear far less frequently than words.




(a) Standard Transformer model, where the user and the item have no contribution to each generation step.

(b) Our PETER model, where the user and item IDs play significant roles in the generation steps.

**Figure 5.1: Attention visualization of two models when generating an explanation for the same user-item pair (see the first two columns). They are both from the last attention layer, so the target sequences are offset by one position for better illustration. The larger the attention weights, the lighter the cells.**

For example, a paragraph of review (and thus hundreds of words) on e-commerce platform only corresponds to a single pair of user ID and item ID. As such, the IDs may be regarded as out-of-vocabulary tokens, to which the Transformer model is insensitive. As shown in Fig. 5.1(a), when generating an explanation for a user-item pair, standard Transformer relies heavily on the special $<bos>$ token instead of the user or the item. This would result in identical explanations over different user-item pairs (see USR score in Tables 5.2, 5.3 and 5.4), making the explanations less personalized.

To address this problem, we bridge IDs and words by designing an elegant task called *context prediction*, which maps IDs onto words to be generated by the explanation task. This in some way resembles one's drafting-polishing process, where by predicting some words the context prediction task does the job of drafting. Then, the explanation generation task polishes these words so as to form a readable sentence. Meanwhile, we demonstrate that conducting recommendation task on the

same model is also feasible, so we name it PETER[1], which stands for PErsonalized Transformer for Explainable Recommendation. As we can see in Fig. 5.1(b), when PETER generates an explanation for the same user-item pair, it can utilize the information of both the user and the item, which illustrates the effectiveness of our context prediction task. In addition, we show that PETER is flexible to accommodate any number of item features that could help guide it to talk about certain topics, e.g., "great jacket, especially for the price" for the feature "price". This makes it more general than the neural template approach introduced in the previous chapter.

In what follows, we first formulate the problem in Section 5.2, and then present our explanation generation method PETER in Section 5.3. Experimental setup and results analysis are respectively given in Sections 5.4 and 5.5. We make a final summary in Section 5.6.

## 5.2 Problem Formulation

The goal of our explanation task is to generate a natural language sentence $\hat{E}_{u,i}$ for a pair of user $u$ and item $i$ to justify why $i$ is recommended to $u$. Meanwhile, our proposed model PETER can also make recommendations by estimating a rating $\hat{r}_{u,i}$ that predicts $u$'s preference towards $i$. At the testing stage, only user $u$ and item $i$ are used as input for producing both explanation and recommendation. When item features $F_{u,i}$ are available, our model is flexible to incorporate them by simply concatenating them at the beginning of the explanation. In this case, the features are also needed in the testing stage. In the following, we discuss both cases.

## 5.3 Model Description

In this section, we present the details of our model PETER. First, we show how to encode different types of tokens in a sequence. Then, we briefly review Transformer

[1]Codes available at https://github.com/lileipisces/PETER

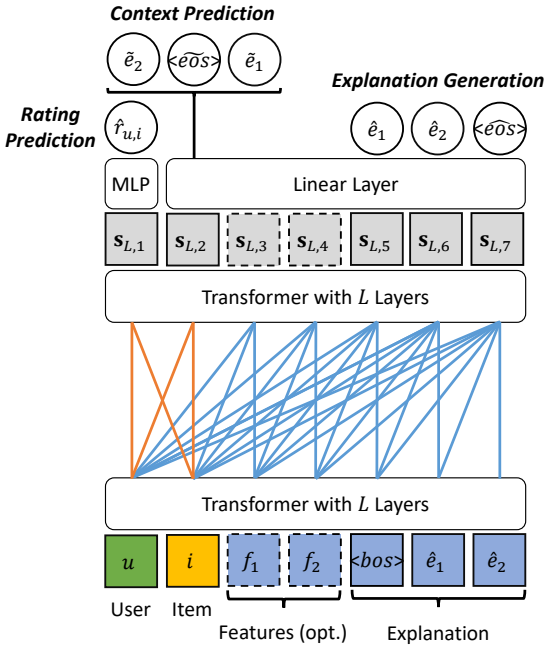



Figure 5.2: Our proposed model PETER that contains three tasks. The input features are optional.

and introduce our revised attention masking matrix. At last, we formulate the three tasks, i.e., explanation generation, context prediction and recommendation, and integrate them into a multi-task learning framework.

### 5.3.1 Input Representation

We first introduce our way to encode heterogeneous input into vector representations. As shown in Fig. 5.2, the input to our model is a sequence, consisting of user ID $u$, item ID $i$, features $F_{u,i}$, and explanation $E_{u,i}$. The user and the item serve for the purpose of personalization, i.e., aiming to make the generated explanation reflect both the user's interests and the item's attributes. The features can guide the model to talk about certain topics. For instance, a conversational recommender system may explain a recommendation's specialty to the user with the goal of knowing more about his/her preference [25]; Or when the user himself/herself proactively asks the recommender system to describe a key feature of a recommendation [64]. Since the features are not always available, in our experiments we test both cases (with and without them). When they are available, the input sequence can be repre-

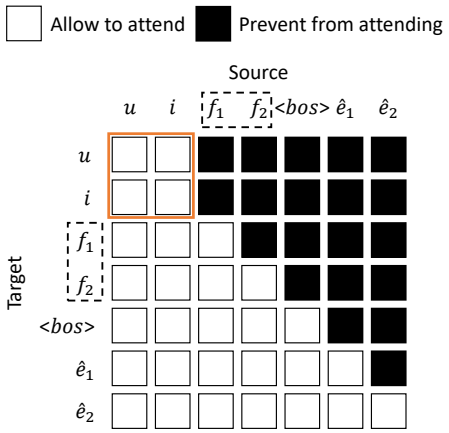



Figure 5.3: The attention masking used in our model that we call PETER masking. The orange box highlights its difference from the Left-to-Right masking.

sented as $S = [u, i, f_1, \cdots, f_{|F_{u,i}|}, e_1, \cdots, e_{|E_{u,i}|}]$, where $f_1, \cdots, f_{|F_{u,i}|}$ are the features and $e_1, \cdots, e_{|E_{u,i}|}$ are the explanation's word sequence. $|F_{u,i}|$ denotes the number of features and $|E_{u,i}|$ is the number of words in the explanation.

Clearly there are three types of tokens in the sequence $S$, i.e., users, items, and words (including features), for which we prepare three sets of randomly initialized token embeddings $\mathbf{U}$, $\mathbf{I}$ and $\mathbf{V}$ respectively, besides the positional embeddings $\mathbf{P}$ that encode the position of each token in the sequence. These embeddings will be updated via back-propagation during the training process. Notice that, we do not add users and items to the vocabulary $\mathcal{V}$, given that it costs more time to predict a word out of the huge amount of IDs (for example, millions of users and items in e-commerce). After performing embedding look-up, we can obtain the sequence's token representation $[\mathbf{u}, \mathbf{i}, \mathbf{f}_1, \cdots, \mathbf{f}_{|F_{u,i}|}, \mathbf{e}_1, \cdots, \mathbf{e}_{|E_{u,i}|}]$ and its positional representation $[\mathbf{p}_1, \cdots, \mathbf{p}_{|S|}]$, where $|S|$ is the length of the sequence. The input representation of the sequence is the addition of the corresponding token representation and positional representation, denoted as $\mathbf{S}_0 = [\mathbf{s}_{0,1}, \cdots, \mathbf{s}_{0,|S|}]$.

### 5.3.2 Transformer and Attention Masking

To enable the three tasks, we show how to modify the attention masking mechanism in Transformer [114]. Transformer consists of $L$ identical layers, each of which

is composed of two sub-layers: multi-head self-attention and position-wise feed-forward network. The $l$-th layer encodes the previous layer's output $\mathbf{S}_{l-1}$ into $\mathbf{S}_l$, where $l \in [1, L]$. In the multi-head self-attention sub-layer, the computation of each attention head is also identical, and among the $H$ heads of the $l$-th layer, the $h$-th head $\mathbf{A}_{l,h}$ is computed as follows:

$$\begin{aligned}
\mathbf{A}_{l,h} &= \text{softmax}(\frac{\mathbf{Q}_{l,h}\mathbf{K}_{l,h}^{\top}}{\sqrt{d}} + \mathbf{M})\mathbf{V}_{l,h} \\
\mathbf{Q}_{l,h} &= \mathbf{S}_{l-1}\mathbf{W}_{l,h}^{Q}, \mathbf{K}_{l,h} = \mathbf{S}_{l-1}\mathbf{W}_{l,h}^{K}, \mathbf{V}_{l,h} = \mathbf{S}_{l-1}\mathbf{W}_{l,h}^{V} \\
\mathbf{M} &= \begin{cases} 0, & \text{Allow to attend} \\ -\infty, & \text{Prevent from attending} \end{cases}
\end{aligned} \tag{5.3.1}$$

where $\mathbf{S}_{l-1} \in \mathbb{R}^{|S| \times d}$ is the $(l-1)$-th layer's output, $\mathbf{W}_{l,h}^{Q}, \mathbf{W}_{l,h}^{K}, \mathbf{W}_{l,h}^{V} \in \mathbb{R}^{d \times \frac{d}{H}}$ are projection matrices, $d$ denotes the dimension of embeddings, and $\mathbf{M} \in \mathbb{R}^{|S| \times |S|}$ is the attention masking matrix.

Each element in $\mathbf{M}$ controls whether a token in the sequence can attend to another. For example, in the unidirectional left-to-right language model [94], the lower triangular part of $\mathbf{M}$ is set to 0 and the remaining part $-\infty$, so as to allow each token to attend to past tokens (including itself), but prevent it from attending to future tokens. We call it *Left-to-Right Masking*. As our model is not limited to the left-to-right explanation generation task, we modify the masking mechanism to accommodate the other two tasks (i.e., context prediction and recommendation). As shown in Fig. 5.3, the first two tokens $u$ and $i$ in the sequence can attend to each other, because both context prediction and recommendation tasks need them. To echo our model, we name it *PETER Masking*.

### 5.3.3 Explanation and Recommendation

In the following, we perform the three tasks, after obtaining the sequence's final representation $\mathbf{S}_L = [\mathbf{s}_{L,1}, \cdots, \mathbf{s}_{L,|S|}]$ from Transformer. The key challenge lies in the personalization of explanation generation task, for which we design the context prediction task. For both tasks, we apply a linear layer to the final representation of

each token to map it onto a $|\mathcal{V}|$-sized vector. As an example, after passing through this layer, $\mathbf{s}_{L,t}$ becomes $\mathbf{c}_t$:

$$\mathbf{c}_t = \text{softmax}(\mathbf{W}^v \mathbf{s}_{L,t} + \mathbf{b}^v) \tag{5.3.2}$$

where $\mathbf{W}^v \in \mathbb{R}^{|\mathcal{V}| \times d}$ and $\mathbf{b}^v \in \mathbb{R}^{|\mathcal{V}|}$ are weight parameters. The vector $\mathbf{c}_t$ represents the probability distribution over the vocabulary $\mathcal{V}$, from which a word $e$ with probability $c_t^e$ can be sampled.

**Explanation Generation:** We adopt the Negative Log-Likelihood (NLL) as the explanation task's loss function, and compute the mean of user-item pairs in the training set:

$$\mathcal{L}_e = \frac{1}{|\mathcal{T}|} \sum_{(u,i) \in \mathcal{T}} \frac{1}{|E_{u,i}|} \sum_{t=1}^{|E_{u,i}|} -\log c_{2+|F_{u,i}|+t}^{e_t} \tag{5.3.3}$$

where $\mathcal{T}$ denotes the training set. The probability $c_t^{e_t}$ is offset by $2 + |F_{u,i}|$ positions because the explanation is placed at the end of the sequence, and $|F_{u,i}| = 0$ when the features are unavailable.

At the testing stage, along with $u$, $i$, and $F_{u,i}$ (if available), we feed the model a special begin-of-sequence token $<bos>$. From its resulting probability distribution $\mathbf{c}_{<bos>}$, the model can predict a word. For simplicity, among the many decoding methods, we opt for greedy decoding that samples the word with the largest probability. Then we can concatenate this predicted word at the end of the sequence to form a new input sequence for generating another word. We do this repeatedly until the model produces a special end-of-sequence token $<eos>$, or the generated explanation $\hat{E}_{u,i}$ reaches a pre-defined length.

**Context Prediction:** As discussed earlier, when there is only one task of explanation generation, Transformer fails to make use of user ID and item ID, resulting in identical sentences. To address this issue, we design this task to map the IDs onto the words in the explanation, so as to build a connection between them. Since the first two positions ($u$ and $i$) of the sequence are allowed to attend to each other, both of their final representations absorb the information of the user and the item.

Thus, we can use either of them to perform this task. Here, we use the 2nd one for better illustration in Fig. 5.2. Again, we adopt NLL as the loss function:

$$\mathcal{L}_c = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} \frac{1}{|E_{u,i}|} \sum_{t=1}^{|E_{u,i}|} -\log c_2^{e_t} \tag{5.3.4}$$

where the words to be predicted are still those in the ground-truth explanation, but they are predicted once for all rather than being autoregressively generated one by one. In more details, these words are resulted from the representation corresponding to $i$, which is why they are not sequentially ordered in Fig. 5.2. This would give the IDs some rudimentary linguistic meanings, so that in the follow-up steps they could be used for explanation generation.

**Rating Prediction:** Recommendation can be seen as a regression problem where the goal is to predict a score $\hat{r}_{u,i}$ based on the IDs of user $u$ and item $i$. As both $u$ and $i$ in the sequence can attend to each other, their final representations capture the interaction between them. Next, we map the 1st representation $\mathbf{s}_{L,1}$ corresponding to $u$ into a scalar (because the 2nd one that corresponds to $i$ is used for context prediction). To this end, we employ multi-layer perceptron (MLP) with one hidden layer as follows:

$$\hat{r}_{u,i} = \mathbf{w}_r^\top \sigma(\mathbf{W}_r \mathbf{s}_{L,1} + \mathbf{b}_r) + b_r \tag{5.3.5}$$

where $\mathbf{W}_r \in \mathbb{R}^{d\times d}$, $\mathbf{b}_r \in \mathbb{R}^d$, $\mathbf{w}_r \in \mathbb{R}^d$ and $b_r \in \mathbb{R}$ are weight parameters, and $\sigma(\cdot)$ is the sigmoid function. Therefore, it can be seen that it is feasible to do both recommendation and explanation on Transformer. For this task, we use Mean Square Error (MSE) as the loss function:

$$\mathcal{L}_r = \frac{1}{|\mathcal{T}|} \sum_{(u,i)\in\mathcal{T}} (r_{u,i} - \hat{r}_{u,i})^2 \tag{5.3.6}$$

where $r_{u,i}$ is the ground-truth rating.

**Multi-task Learning:** At last, we integrate the three tasks into a multi-task learning framework whose objective function is defined as:

$$\mathcal{J} = \min_{\Theta} (\lambda_e \mathcal{L}_e + \lambda_c \mathcal{L}_c + \lambda_r \mathcal{L}_r) \tag{5.3.7}$$

Table 5.1: Statistics of the three datasets.

| | Yelp | Amazon | TripAdvisor |
|---|---|---|---|
| #users | 27,147 | 7,506 | 9,765 |
| #items | 20,266 | 7,360 | 6,280 |
| #records | 1,293,247 | 441,783 | 320,023 |
| #features | 7,340 | 5,399 | 5,069 |
| #records / user | 47.64 | 58.86 | 32.77 |
| #records / item | 63.81 | 60.02 | 50.96 |
| #words / explanation | 12.32 | 14.14 | 13.01 |

where $\Theta$ denotes all the trainable parameters in the model, and $\lambda_e$, $\lambda_c$ and $\lambda_r$ are regularization coefficients that balance the learning of different tasks. In this way, the model can be trained efficiently in an end-to-end manner.

## 5.4 Experimental Setup

### 5.4.1 Datasets

For experimentation, we adopt three publicly available explainable recommendation datasets, as well as their data splits[2] [64]. During the splitting process, each dataset is randomly divided into training, validation and testing sets with ratio 8:1:1 for 5 times, and the training set holds at least one record for each user and each item. The three datasets are respectively from TripAdvisor (hotel), Amazon (movies & TV) and Yelp (restaurant). Each record in the datasets is comprised of a user ID, an item ID, a rating, an explanation, and a feature. The explanations are sentences extracted from user reviews. Each explanation contains at least one item feature, e.g., "bedroom", which ensures the explanation quality. Statistics of the datasets are shown in Table 5.1. We can see that Yelp is much larger than the other two in

[2]https://github.com/lileipisces/NETE

terms of size, making it closer to the real-world situation where there are millions of users and items.

### 5.4.2 Evaluation Metrics

To evaluate the recommendation performance, we adopt two commonly used metrics: Root Mean Square Error (**RMSE**) and Mean Absolute Error (**MAE**). As to explanation performance, we measure the generated explanations from two main perspectives: text quality and explainability. For the former, we adopt **BLEU** [91] in machine translation and **ROUGE** [75] in text summarization, and report BLEU-1 and BLEU-4, and Precision, Recall and F1 of ROUGE-1 and ROUGE-2. Though being widely used, BLUE and ROUGE are not flawless. For example, it is difficult for them to detect the problem of identical sentences as generated by standard Transformer. These sentences could not be used as explanations, because they are less likely to well explain the special property of different recommendations. To quantitatively measure how severe the problem is, we adopt **USR** that computes the Unique Sentence Ratio of generated explanations [64].

Text quality, however, is not equal to explainbility. In the case of explainable recommendation, users may value more an explanation that justifies a recommendation's advantages on certain features [64, 16]. To this end, we adopt the other three metrics proposed in [64]: Feature Matching Ratio (**FMR**), Feature Coverage Ratio (**FCR**) and Feature Diversity (**DIV**). FMR measures whether a generated explanation contains the feature in the ground-truth. FCR is computed as the number of distinct features contained in all the generated explanations, divided by the total number of features in the whole dataset. DIV measures the intersection of features between any two generated explanations.

For RMSE, MAE and DIV, the lower, the better, while it is opposite for the rest of metrics.

### 5.4.3 Compared Methods

We introduce baselines, first for explanation and then for recommendation. For the former, we divide the baselines into two groups, depending on whether the feature is used or not. The following models leverage only user and item IDs to generate explanations. We denote our model without feature as PETER.

- **Transformer** [114] performs the explanation generation task by treating user and item IDs as words. We also tested encoder-decoder Transformer, where the encoder encodes the IDs for the decoder to decode, but its results turned out to be the same, so we do not report it.

- **NRT** [70] can predict a rating and generate a tip simultaneously based on user and item IDs. We take the explanations in the datasets as tips. Moreover, we found that the model's problem of generating identical sentences (as discussed in [64]) is caused by the L2 regularization in its original design[3]. For fair comparison, we removed it.

- **Att2Seq** [31] is a review generation approach and we take the explanations as reviews. This model has an attention module, but we found that it makes the generated text unreadable. To be fair, we removed it as well.

When features are used, we denote our model as PETER+, and compare it with two recent models:

- **ACMLM** [90] is a fine-tuned BERT [30], where an attention layer is introduced to encode the features from both the user and the item. By predicting masked tokens, this model can produce diverse sentences.

- **NETE** [64] is a tailored GRU [27] that incorporates a given feature into the decoding process to generate template-like explanations. It can also make recommendations.

[3]We found out that the L2 regularization in NRT's implementation is actually $\lambda||\Theta||$, but $\lambda||\Theta||^2$ was reported, where $\Theta$ denotes all model parameters and $\lambda = 10^{-4}$.

For recommendation, besides NRT and NETE, we include another two traditional methods:

- **PMF** [88] is a standard probabilistic matrix factorization method that characterizes users and items by latent factors.

- **SVD++** [56] leverages a user's interacted items to enhance the latent factors.

### 5.4.4 Implementation Details

We train each model on the training set, tune the hyper-parameters on the validation set, and report the performance on the testing set. The results are averaged on the 5 data splits. We adopt the codes of ACMLM and NETE, and implement all the other methods. For NRT, Att2Seq, NETE and our PETER and PETER+, we set the size of vocabulary to 20,000 by keeping the most frequent words. We do not apply this to Transformer, otherwise users and items (regarded as words) may be filtered out. We set both the number of context words and the length of explanations to 15, because the mean length of explanations is approximately 13 (see Table 5.1). ACMLM adopts sub-words, so we do not apply the above two steps to it. We reuse the other default settings of the baselines.

For Transformer, PETER and PETER+, we set the embedding size $d$ to 512 and the dimension of feed-forward network to 2,048, following [114], but the number of layers $L$ and attention heads $H$ are both 2. For our models PETER and PETER+, we set the regularization coefficients $\lambda_e$, $\lambda_c$ and $\lambda_r$ to 1.0, 1.0 and 0.1, respectively. We optimize the model via stochastic gradient descent [102], and apply gradient clipping [92] with a threshold of 1.0. The batch size is set to 128, and the learning rate 1.0. At each epoch, we save the model if it achieves the lowest loss on the validation set, but when there is no improvement, we decrease the learning rate by a factor of 0.25. When the latter happens for 5 times, we stop training and load the saved model for prediction.

Table 5.2: Performance comparison of natural language generation methods in terms of Explainability and Text Quality on Yelp dataset. The methods are divided into two groups according to whether features are used or not. B1 and B4 stand for BLEU-1 and BLEU-4. R1-P, R1-R, R1-F, R2-P, R2-R and R2-F denote Precision, Recall and F1 of ROUGE-1 and ROUGE-2. BLEU and ROUGE are percentage values (% symbol omitted for table clarity), while the others are absolute values. The best performing values are boldfaced, and the second best underlined. ** indicates the statistical significance over the second best baseline for $p < 0.01$ via Student's t-test.

| | Explainability | | | Text Quality | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FMR↑ | FCR↑ | DIV↓ | USR↑ | B1↑ | B4↑ | R1-P↑ | R1-R↑ | R1-F↑ | R2-P↑ | R2-R↑ | R2-F↑ |
| Transformer | 0.06 | 0.06 | 2.46 | 0.01 | 7.39 | 0.42 | **19.18** | 10.29 | 12.56 | 1.71 | 0.92 | 1.09 |
| NRT | <u>0.07</u> | 0.11 | <u>2.37</u> | <u>0.12</u> | **11.66** | <u>0.65</u> | 17.69 | <u>12.11</u> | <u>13.55</u> | 1.76 | <u>1.22</u> | <u>1.33</u> |
| Att2Seq | <u>0.07</u> | <u>0.12</u> | 2.41 | **0.13** | 10.29 | 0.58 | <u>18.73</u> | 11.28 | 13.29 | <u>1.85</u> | 1.14 | 1.31 |
| PETER | **0.08**** | **0.19**** | **1.54**** | **0.13** | <u>10.77</u> | **0.73**** | 18.54 | **12.20** | **13.77**** | **2.02**** | **1.38**** | **1.49**** |
| ACMLM | 0.05 | <u>0.31</u> | **0.95** | **0.95** | 7.01 | 0.24 | 7.89 | 7.54 | 6.82 | 0.44 | 0.48 | 0.39 |
| NETE | <u>0.80</u> | 0.27 | 1.48 | <u>0.52</u> | <u>19.31</u> | <u>2.69</u> | <u>33.98</u> | <u>22.51</u> | <u>25.56</u> | <u>8.93</u> | <u>5.54</u> | <u>6.33</u> |
| PETER+ | **0.86**** | **0.38**** | <u>1.08</u> | 0.34 | **20.80**** | **3.43**** | **35.44**** | **26.12**** | **27.95**** | **10.65**** | **7.44**** | **7.94**** |

Table 5.3: Performance comparison of natural language generation methods in terms of Explainability and Text Quality on Amazon dataset. The methods are divided into two groups according to whether features are used or not. B1 and B4 stand for BLEU-1 and BLEU-4. R1-P, R1-R, R1-F, R2-P, R2-R and R2-F denote Precision, Recall and F1 of ROUGE-1 and ROUGE-2. BLEU and ROUGE are percentage values (% symbol omitted for table clarity), while the others are absolute values. The best performing values are boldfaced, and the second best underlined. ** indicates the statistical significance over the second best baseline for $p < 0.01$ via Student's t-test.

| | Explainability | | | Text Quality | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FMR↑ | FCR↑ | DIV↓ | USR↑ | B1↑ | B4↑ | R1-P↑ | R1-R↑ | R1-F↑ | R2-P↑ | R2-R↑ | R2-F↑ |
| Transformer | 0.10 | 0.01 | 3.26 | 0.00 | 9.71 | 0.59 | 19.68 | 11.94 | 14.11 | 2.10 | 1.39 | 1.55 |
| NRT | **0.12** | 0.07 | 2.93 | 0.17 | **12.93** | 0.96 | **21.03** | 13.57 | **15.56** | 2.71 | 1.84 | 2.05 |
| Att2Seq | **0.12** | 0.20 | 2.74 | **0.33** | 12.56 | 0.95 | 20.79 | 13.31 | 15.35 | 2.62 | 1.78 | 1.99 |
| PETER | **0.12**** | **0.21** | **1.75**** | 0.29 | 12.77 | **1.17**** | 19.81 | **13.80** | 15.23 | **2.80** | **2.08**** | **2.20**** |
| ACMLM | 0.10 | **0.31** | 2.07 | **0.96** | 9.52 | 0.22 | 11.65 | 10.39 | 9.69 | 0.71 | 0.81 | 0.64 |
| NETE | 0.71 | 0.19 | 1.93 | 0.57 | 18.76 | 2.46 | 33.87 | 21.43 | 24.81 | 7.58 | 4.77 | 5.46 |
| PETER+ | **0.77**** | **0.31**** | **1.20**** | 0.46 | **19.75**** | **3.06**** | **34.71**** | **23.99**** | **26.35**** | **9.04**** | **6.23**** | **6.71**** |

Table 5.4: Performance comparison of natural language generation methods in terms of Explainability and Text Quality on TripAdvisor dataset. The methods are divided into two groups according to whether features are used or not. B1 and B4 stand for BLEU-1 and BLEU-4. R1-P, R1-R, R1-F, R2-P, R2-R and R2-F denote Precision, Recall and F1 of ROUGE-1 and ROUGE-2. BLEU and ROUGE are percentage values (% symbol omitted for table clarity), while the others are absolute values. The best performing values are boldfaced, and the second best underlined. ** and * indicate the statistical significance over the second best baseline respectively for $p < 0.01$ and $p < 0.05$ via Student's t-test.

| | Explainability | | | Text Quality | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | FMR↑ | FCR↑ | DIV↓ | USR↑ | B1↑ | B4↑ | R1-P↑ | R1-R↑ | R1-F↑ | R2-P↑ | R2-R↑ | R2-F↑ |
| Transformer | 0.04 | 0.00 | 10.00 | 0.00 | 12.79 | 0.71 | 16.52 | **16.38** | 15.88 | 2.22 | **2.63** | **2.34** |
| NRT | 0.06 | 0.09 | 4.27 | 0.08 | 15.05 | 0.99 | 18.22 | 14.39 | 15.40 | 2.29 | 1.98 | 2.01 |
| Att2Seq | 0.06 | **0.15** | 4.32 | **0.17** | 15.27 | 1.03 | 18.97 | 14.72 | 15.92 | **2.40** | 2.03 | 2.09 |
| PETER | **0.07**** | 0.13 | **2.95**** | 0.08 | **15.96**** | **1.11*** | **19.07** | 16.09 | **16.48**** | 2.33 | 2.17 | 2.09 |
| ACMLM | 0.07 | **0.41** | **0.78** | **0.94** | 3.45 | 0.02 | 4.86 | 3.82 | 3.72 | 0.18 | 0.20 | 0.16 |
| NETE | 0.78 | 0.27 | 2.22 | 0.57 | 22.39 | 3.66 | 35.68 | 24.86 | 27.71 | 10.20 | 6.98 | 7.66 |
| PETER+ | **0.89**** | 0.35 | 1.61 | 0.25 | **24.32**** | **4.55**** | **37.48**** | **29.21**** | **30.49**** | **11.92**** | **8.98**** | **9.24**** |

## 5.5 Results and Analysis

### 5.5.1 Quantitative Analysis on Explanations

In Tables 5.2, 5.3 and 5.4, we compare the performance of explanation generation methods in two groups. We first analyze models that make use of item features (i.e., ACMLM, NETE and PETER+). Our PETER+ consistently and significantly outperforms ACMLM and NETE on the three datasets in terms of text quality (BLEU and ROUGE). This shows the effectiveness of our model in generating high-quality sentences. Notice that a user survey was conducted in [62] and reported that NETE's explanations were perceived useful by most participants. It suggests that our model's explanations with better quality could also be very useful to real users.

Again, in terms of text quality, the performance gap between PETER+ and ACMLM (a fine-tuned BERT) is extremely large, because the latter's generation is achieved by predicting masked tokens, which is quite different from auto-regressive generation. This may explain why ACMLM produces diverse sentences (high USR), which, however, is less meaningful when text quality cannot be guaranteed. Furthermore, PETER+ beats both ACMLM and NETE on the explainability metric FMR that cares about whether a generated explanation mentions the feature in the ground-truth. This is quite useful in real-world applications when the system is asked to explain a particular feature. Regarding the other two explainability metrics FCR and DIV, PETER+ is also very competitive. ACMLM gains better performance on some cases, because at the training stage it is exposed to more features (from both the user and the item), which is unfair to both PETER+ and NETE.

Next, we discuss the results of the models that only leverage user and item IDs for generation. As it can be seen, Transformer generates identical explanations on each dataset, resulting in nearly 0 score on Unique Sentence Ratio (USR). Owing to the context prediction task, our PETER successfully addresses this issue, producing diverse (comparable USR) and high-quality (best BLEU-4) sentences. In particular, on the largest dataset Yelp, it achieves the best performance on most of the metrics.

Table 5.5: Efficiency comparison of two Transformer-based models in terms of training minutes on the TripAdvisor dataset, tested on NVIDIA Tesla P40.

| | Time | Epochs | Time/Epoch |
|---|---|---|---|
| ACMLM | 97.0 | **3** | 32.3 |
| PETER+ | **57.7** | 25 | **2.3** |

This again demonstrates the effectiveness of our model. On Amazon and TripAdvisor, NRT and Att2Seq are very competitive, because we fixed their generation issues (see Section 5.4.3). In addition, the two datasets are small and thus the training samples are limited, so our model may underfit, which is why it does not always reach the best performance.

Besides explanation performance, we also investigate the efficiency of different Transformer-based models. On the same machine (NVIDIA Tesla P40) and dataset (TripAdvisor), we compare the training minutes of ACMLM and our PETER+ in Table 5.5. Compared with ACMLM, our model takes less time to train (2.3 minutes per epoch), since it has only 2 layers and thus less parameters. But because it is unpretrained and learned from scratch, it needs more training epochs.

### 5.5.2 Qualitative Case Study on Explanations

In Table 5.6, we present two examples generated by PETER and PETER+ on the TripAdvisor dataset. We can see that PETER generates distinct context words and explanations for different user-item pairs. This confirms that our proposed solution can indeed endow the user and item IDs with linguistic meanings, as well as achieving certain degree of personalization for natural language generation. Among the commonly used context words, e.g., "the", there are some important features (underlined), according to which the model then generates an explanation that talks about them. Admittedly, there is still much room for improvement of the context prediction task, so as to more accurately predict the features in the ground-truth

Table 5.6: Context words and explanations on two different cases as generated by our PETER and PETER+ on TripAdvisor dataset. The boldfaced words in the ground-truth are the key features. Generated features are underlined.

| | Top-15 Context Words | Explanation |
|---|---|---|
| Ground-truth | | the **rooms** are spacious and the bathroom has a large tub |
| PETER | <eos> the and a pool was with nice is very were to good in of | the pool area is nice and the gym is very well equipped <eos> |
| PETER+ | <eos> the and a was pool with to nice good very were is of in | the rooms were clean and comfortable <eos> |
| Ground-truth | | beautiful **lobby** and nice bar |
| PETER | <eos> the and a was were separate bathroom with shower large very had in is | the bathroom was large and the shower was great <eos> |
| PETER+ | <eos> the and a was bathroom shower with large in separate were room very is | the lobby was very nice and the rooms were very comfortable <eos> |

(e.g., "rooms" vs. "pool" in the first example). One alternative is to leverage the features to guide the model's generation. This explains why PETER+ is able to generate an explanation that talks about "rooms" rather than "pool", making it semantically closer to the ground-truth. It thus demonstrates our model's flexibility in incorporating these features.

### 5.5.3 Recommendation Performance

Table 5.7 presents the performance comparison of different recommendation methods. On the largest dataset Yelp with approximately 1.3 million records, our model

Table 5.7: Recommendation performance comparison in terms of RMSE and MAE. The best performing values are boldfaced.

| | Yelp | | Amazon | | TripAdvisor | |
|---|---|---|---|---|---|---|
| | RMSE↓ | MAE↓ | RMSE↓ | MAE↓ | RMSE↓ | MAE↓ |
| PMF | 1.0856 | 0.8765 | 1.0344 | 0.8072 | 0.8703 | 0.6961 |
| SVD++ | 1.0112 | 0.7845 | 0.9649 | 0.7185 | 0.7981 | 0.6095 |
| NRT | 1.0123 | 0.7842 | **0.9499** | **0.7046** | 0.7918 | 0.6094 |
| NETE | **1.0097** | 0.7889 | 0.9608 | 0.7269 | **0.7916** | **0.6079** |
| PETER | 1.0130 | **0.7839** | 0.9533 | 0.7056 | 0.8073 | 0.6253 |

PETER performs as good as the three competitive baselines (i.e., SVD++, NRT and NETE), which shows the rationale of our recommendation module. Since our model PETER has more parameters to learn, it may underfit on small datasets. This explains why it does not always perform the best on TripAdvisor and Amazon. When more training data are available to Transformer, usually the performance will become better, as evidenced by GPT-2 [95] and GPT-3 [10]. Thus, we can expect our model to perform well in real-world applications, where the training data are bigger than the testing datasets, e.g., billion-scale users in Amazon.

### 5.5.4 Ablation Study

In Table 5.8, we provide an ablation study conducted on the TripAdvisor dataset. After disabling the context prediction task $\mathcal{L}_c$ by setting $\lambda_c = 0$, the performances of both explainability and text quality drop dramatically, and the unique sentence ratio (USR) is nearly approaching Transformer's (see Tables 5.2, 5.3 and 5.4). It hence confirms this task's effectiveness. As $\mathcal{L}_c$ is highly correlated with the recommendation task $\mathcal{L}_r$ via the user and item IDs (see Section 5.3.3), the removal of $\mathcal{L}_c$ leads to slight improvement on recommendation performance. We can also observe a reversed phenomenon when we disable $\mathcal{L}_r$. When PETER masking is replaced by

Table 5.8: Ablation study on the smallest dataset TripAdvisor. Arrows ↑ and ↓ respectively denote the performance increase and decrease compared with PETER.

| | Explainability | | | Text Quality | | | Recommendation | |
|---|---|---|---|---|---|---|---|---|
| | FMR | FCR | DIV | USR | BLEU-1 | BLEU-4 | RMSE | MAE |
| Disable $\mathcal{L}_c$ | 0.06 ↓ | 0.03 ↓ | 5.75 ↓ | 0.01 ↓ | 15.37 ↓ | 0.86 ↓ | 0.80 ↑ | 0.61 ↑ |
| Disable $\mathcal{L}_r$ | 0.07 | 0.14 ↑ | 2.90 ↑ | 0.10 ↑ | 16.16 ↑ | 1.15 ↑ | 3.23 ↓ | 3.10 ↓ |
| Left-to-Right Masking | 0.07 | 0.15 ↑ | 2.68 ↑ | 0.12 ↑ | 15.73 ↓ | 1.11 | 0.87 ↓ | 0.68 ↓ |
| PETER | 0.07 | 0.13 | 2.95 | 0.08 | 15.96 | 1.11 | 0.81 | 0.63 |

the Left-to-Right masking that prevents the model from accessing the item information, the recommendation performance drops sharply. Overall, PETER reaches an optimal situation, where its explainability, text quality and recommendation performance are all reasonably good.

## 5.6 Summary

In this chapter, we propose a simple and effective solution to address the personalized generation problem of Transformer, unleashing its language modeling power to generate explanations for recommender systems. Extensive experiments show that the solution is both effective and efficient. It opens up a new way of exploiting Transformer by designing good tasks instead of scaling up model size. There are various applications of personalized generation for which Transformer is still less explored, e.g., personalized conversational agents. It is also promising to incorporate item images into the model, so as to generate visual explanations for recommendations, since "a picture is worth a thousand words". Another meaningful extension is to adapt the model to cross-lingual explanation generation, because international platforms, e.g., Amazon, may serve users who speak different languages.

# Chapter 6

# Explanation Ranking

## 6.1 Background

In the proceeding two chapters, we introduced two natural language explanation generation approaches. Although they can produce high-quality explanations, this type of approach is not flawless. For example, in our previous experiments [64, 67] we observe that a large amount of generated explanations are commonly seen sentences in the training data, e.g., "the food is good" as an explanation for a recommended restaurant. This means that the models are fitting the given samples rather than creating new sentences. Moreover, even strong language models such as Transformer [114] trained on a large text corpus may generate contents that deviate from facts, e.g., "four-horned unicorns" [95].

To tackle this problem, we wonder whether explanations could be ranked, just as information retrieval systems that assume the available contents (e.g., documents or images) are correct, and intelligently rank them according to a given query. The basic idea of explanation ranking is to train a model that can select appropriate explanations from an explanation pool for a recommendation, which is different from natural language explanation generation approaches that generate explanations by learning the pattern from training data. The ranking formulation also makes the standard evaluation of explanations possible, via ranking metrics, such as NDCG, Precision and Recall. Moreover, the ranking formulation can accommodate various


Figure 6.1: Three user reviews for different restaurants from Yelp. Sentences that can be used as explanations are highlighted in colors. Co-occurring explanations across different reviews are highlighted in rectangles.

explanation styles, such as sentences, images, and even new styles yet to be invented, as long as user-item-explanation interactions are available.

However, since such interactions are usually unavailable in existing recommender systems, the key challenge then becomes how to create them. As inspired by the wisdom of the crowd, our solution is to extract co-occurring components that can be used as explanations from user-item interactions. As an instantiation, we extract sentences (in rectangles in Fig. 6.1) across different user reviews, because they reflect users' authentic evaluation towards items. By finding the commonly used sentences, the quality of explanations such as readability and expressiveness can be guaranteed. A follow-up problem is how to efficiently detect the nearly identical sentences across reviews in a dataset. Computing the similarity between any two sentences in a dataset is feasible but less efficient, since it has a quadratic time complexity. To make this process more efficient, we develop a method that can categorize sentences into different groups, based on Locality Sensitive Hashing (LSH) [96] which is devised for near-duplicates detection. We create three benchmark datasets, and name them EXTRA[1], which stands for EXplanaTion RAnking.

With the evaluation and data, we further investigate the potential impacts of explanations, such as higher chance of item click, conversion or fairness [108], which

[1]Codes and datasets available at https://github.com/lileipisces/EXTRA

are less explored but are particularly important in commercial systems. Without an appropriate approach to explanation evaluation, explanations have usually been modeled as an auxiliary function of the recommendation task in most explainable models [133, 14, 105, 80, 24]. A recent study [23] shows that fine-tuning the parallel task of feature ranking can boost the recommendation performance. Moreover, it has been shown in a user study that users' feedback on explanation items could help to improve recommendation performance [39]. Based on these findings, we design an item-explanation joint-ranking framework to study if showing some particular explanations could lead to increased item acceptance rate, i.e., improving the recommendation performance. Furthermore, we are motivated to identify how the recommendation task and the explanation task would interact with each other, whether there is a trade-off between them, and how to achieve the most ideal solution for both.

However, the above investigation cannot proceed without addressing the inherent data sparsity issue in the user-item-explanation interactions. In traditional pair-wise data, each user may be associated with several items, but in the user-item-explanation triplets data, each user-item pair may be associated with only one explanation. In consequence, the data sparsity problem is severer for explanation ranking. Therefore, how to design an effective model for such one-shot learning scenario becomes a great challenge. Our solution is to separate user-item-explanation triplets into user-explanation and item-explanation pairs, which significantly alleviates the data sparsity problem. Based on this idea, we design two types of model[2]. First, a general model that only makes use of IDs, aims to accommodate a variety of explanation styles, such as sentences and images. Second, a domain-specific model based on BERT [30] further leverages the semantic features of the explanations to enhance the ranking performance.

In the following, we first formulate the problems in Section 6.2. Then, our proposed models and the joint-ranking framework are presented in Section 6.3. The

[2]Codes available at https://github.com/lileipisces/BPER

data creation approach is given in Section 6.4. Section 6.5 introduces the experimental setup, and the discussion of results is provided in Section 6.6. We summarize this chapter in Section 6.7.

## 6.2 Problem Formulation

The key notations and concepts for the problems are presented in Table 6.1. We use $\mathcal{U}$ to denote the set of all users, $\mathcal{I}$ the set of all items and $\mathcal{E}$ the set of all explanations. Then the historical interaction set is given by $\mathcal{T} \subseteq \mathcal{U} \times \mathcal{I} \times \mathcal{E}$. In the following, we first introduce item ranking and explanation ranking respectively, and then the item-explanation joint-ranking.

### 6.2.1 Item Ranking

Personalized recommendation aims at providing a user with a ranked list of items that he/she never interacted with before. For each user $u \in \mathcal{U}$, the list of $M$ items can be generated as follows,

$$\text{Top}(u, M) := \underset{i \in \mathcal{I}/\mathcal{I}_u}{\arg\max^{M}} \hat{r}_{u,\underline{i}} \tag{6.2.1}$$

where $\hat{r}_{u,\underline{i}}$ is the predicted score for a user $u$ on item $i$, and $\mathcal{I}/\mathcal{I}_u$ denotes the set of items on which user $u$ has no interactions. In Eq. (6.2.1), $i$ is underlined, which means that we aim to rank the items.

### 6.2.2 Explanation Ranking

Meanwhile, explanation ranking is the task of finding a list of appropriate explanations for a user-item pair to justify why the recommendation is made. Formally, given a user $u \in \mathcal{U}$ and an item $i \in \mathcal{I}$, the goal of this task is to rank the entire collection of explanations $\mathcal{E}$, and then select the top $N$ to reason why item $i$ is recommended. Specifically, we define this list of top $N$ explanations as:

$$\text{Top}(u, i, N) := \underset{e \in \mathcal{E}}{\arg\max^{N}} \hat{r}_{u,i,\underline{e}} \tag{6.2.2}$$

Table 6.1: Key notations and concepts.

| Symbol | Description |
|---|---|
| $\mathcal{T}$ | training set |
| $\mathcal{U}$ | set of users |
| $\mathcal{I}$ | set of items |
| $\mathcal{I}_u$ | set of items that user $u$ preferred |
| $\mathcal{E}$ | set of explanations |
| $\mathcal{E}_u$ | set of user $u$'s explanations |
| $\mathcal{E}_i$ | set of item $i$'s explanations |
| $\mathcal{E}_{u,i}$ | set of explanations that user $u$ preferred w.r.t. item $i$ |
| $\mathbf{P}$ | latent factor matrix for users |
| $\mathbf{Q}$ | latent factor matrix for items |
| $\mathbf{O}$ | latent factor matrix for explanations |
| $\mathbf{p}_u$ | latent factors of user $u$ |
| $\mathbf{q}_i$ | latent factors of item $i$ |
| $\mathbf{o}_e$ | latent factors of explanation $e$ |
| $b_i$ | bias term of item $i$ |
| $b_e$ | bias term of explanation $e$ |
| $d$ | dimension of latent factors |
| $\alpha$, $\lambda$ | regularization coefficient |
| $\gamma$ | learning rate |
| $T$ | iteration number |
| $M$ | number of recommendations for each user |
| $N$ | number of explanations for each recommendation |
| $\hat{r}_{u,i}$ | score predicted for user $u$ on item $i$ |
| $\hat{r}_{u,i,e}$ | score predicted for user $u$ on explanation $e$ of item $i$ |

where $\hat{r}_{u,i,\underline{e}}$ is the estimated score of explanation $e$ for a given user-item pair $(u, i)$, which could be given by a recommendation model or by the user's true behavior.

### 6.2.3 Item-Explanation Joint-Ranking

The preceding two tasks solely rank either items or explanations. In this task, we further investigate whether it is possible to find an ideal item-explanation pair for a user, to whom the explanation best justifies the item that he/she likes the most. To this end, we treat each item-explanation pair as a joint unit, and then rank these units. Specifically, for each user $u \in \mathcal{U}$, a ranked list of $M$ item-explanation pairs can be produced as follows,

$$\text{Top}(u, M) := \underset{i \in \mathcal{I}/\mathcal{I}_u, e \in \mathcal{E}}{\arg\max^{M}} \hat{r}_{u,\underline{i,e}} \tag{6.2.3}$$

where $\hat{r}_{u,\underline{i,e}}$ is the predicted score for a given user $u$ on the item-explanation pair ($i$, $e$).

We see that either item ranking task or explanation ranking task is a special case of this item-explanation joint-ranking task. Concretely, Eq. (6.2.3) degenerates to Eq. (6.2.1) when explanation $e$ is fixed, while it reduces to Eq. (6.2.2) if item $i$ is already known.

## 6.3 Framework Description

### 6.3.1 Joint-Ranking Reformulation

Suppose we have an ideal model that can perform the aforementioned joint-ranking task. During the prediction stage as in Eq. (6.2.3), there would be $|\mathcal{I}| \times |\mathcal{E}|$ candidate item-explanation pairs to rank for each user $u \in \mathcal{U}$. The runtime complexity is then $O(|\mathcal{U}| \cdot |\mathcal{I}| \cdot |\mathcal{E}|)$, which makes this task impractical, compared with the traditional recommendation task's $O(|\mathcal{U}| \cdot |\mathcal{I}|)$ complexity.

To reduce the complexity, we reformulate the joint-ranking task by performing ranking for items and explanations simultaneously but separately. In this way, we

are also able to investigate whether item ranking and explanation ranking could influence each other, e.g., improving the performance of both. Specifically, during the testing stage, we first follow Eq. (6.2.1) to rank items for each user $u \in \mathcal{U}$, which has the runtime complexity of $O(|\mathcal{U}| \cdot |\mathcal{I}|)$. After that, for $M$ recommendations for each user, we rank and select explanations to justify each of them according to Eq. (6.2.2). The second step's complexity is $O(|\mathcal{U}| \cdot M \cdot |\mathcal{E}|)$, but since $M$ is a constant and $|\mathcal{E}| \ll |\mathcal{I}|$ (see Table 6.2), the overall complexity of the two steps is $O(|\mathcal{U}| \cdot |\mathcal{I}|)$.

In the following, we first analyze the drawback of a conventional Tensor Factorization (TF) model, when it is applied to the explanation ranking problem, and then introduce our solution BPER. Second, we show how to further enhance BPER by utilizing the semantic features of textual explanations (denoted as BPER+). Third, we illustrate their relation to two typical TF methods CD and PITF. At last, we integrate the explanation ranking with item ranking into a multi-task learning framework as a joint-ranking task.

### 6.3.2 Bayesian Personalized Explanation Ranking (BPER)

To perform explanation ranking, the score $\hat{r}_{u,i,e}$ on each explanation $e \in \mathcal{E}$ for a given user-item pair $(u, i)$ must be estimated. As the user-item-explanation ternary relations $\mathcal{T} = \{(u, i, e)|u \in \mathcal{U}, i \in \mathcal{I}, e \in \mathcal{E}\}$ form an interaction cube, we are inspired to employ factorization models to predict the scores. There are a number of tensor factorization techniques, such as Tucker Decomposition (TD) [113], Canonical Decomposition (CD) [12] and High Order Singular Value Decomposition (HOSVD) [29]. Intuitively, one would adopt CD, because of its linear runtime complexity in terms of both training and prediction [99] and its close relation to Matrix Factorization (MF) [88] that has been extensively studied in recent years for item recommendation. Formally, according to CD, the score $\hat{r}_{u,i,e}$ of user $u$ on item $i$'s explanation $e$ can be estimated by the sum over the element-wise multiplication of

the user's latent factors $\mathbf{p}_u$, the item's $\mathbf{q}_i$ and the explanation's $\mathbf{o}_e$:

$$\hat{r}_{u,i,e} = (\mathbf{p}_u \odot \mathbf{q}_i)^\top \mathbf{o}_e = \sum_{k=1}^{d} p_{u,k} \cdot q_{i,k} \cdot o_{e,k} \tag{6.3.4}$$

where $\odot$ denotes the element-wise multiplication of two vectors.

However, this method may not be effective enough due to the inherent sparsity problem of the ternary data. Since each user-item pair $(u, i)$ in the training set $\mathcal{T}$ is unlikely to have interactions with many explanations in $\mathcal{E}$, the data sparsity problem for explanation ranking is severer than that for item recommendation. Simply multiplying the three vectors would hurt the performance of explanation ranking, which is evidenced by our experimental results in Section 6.6.

To mitigate such an issue and to improve the effectiveness of explanation ranking, we propose to separately estimate the user $u$'s preference score $\hat{r}_{u,e}$ on explanation $e$ and the item $i$'s appropriateness score $\hat{r}_{i,e}$ for explanation $e$. To this end, we perform two sets of matrix factorization, rather than employing one single TF model. In this way, the sparsity problem would be considerably alleviated, since the data are reduced to two collections of binary relations, both of which are similar to the case of item recommendation. At last, the two scores $\hat{r}_{u,e}$ and $\hat{r}_{i,e}$ are combined linearly through a hyper-parameter $\mu$. Specifically, the score of user $u$ for item $i$ on explanation $e$ is predicted as follows,

$$\begin{cases} \hat{r}_{u,e} = \mathbf{p}_u^\top \mathbf{o}_e^U + b_e^U = \sum_{k=1}^{d} p_{u,k} \cdot o_{e,k}^U + b_e^U \\ \hat{r}_{i,e} = \mathbf{q}_i^\top \mathbf{o}_e^I + b_e^I = \sum_{k=1}^{d} q_{i,k} \cdot o_{e,k}^I + b_e^I \\ \hat{r}_{u,i,e} = \mu \cdot \hat{r}_{u,e} + (1 - \mu) \cdot \hat{r}_{i,e} \end{cases} \tag{6.3.5}$$

where $\{\mathbf{o}_e^U, b_e^U\}$ and $\{\mathbf{o}_e^I, b_e^I\}$ are two different sets of latent factors for explanations, corresponding to users and items respectively.

Since selecting explanations that are likely to be perceived helpful by users is inherently a ranking-oriented task, directly modeling the relative order of explanations is thus more effective than simply predicting their absolute scores. The Bayesian Personalized Ranking (BPR) criterion [98] meets such an optimization requirement. Intuitively, a user would be more likely to appreciate explanations that

cater to his/her own preferences, while those that do not fit one's interests would be less attractive to the user. Similarly, some explanations might be more suitable to describe certain items, while other explanations might not. To model such type of pair-wise preferences, we compute the difference between two explanations for both user $u$ and item $i$ as follows,

$$\begin{cases} \hat{r}_{u,ee'} = \hat{r}_{u,e} - \hat{r}_{u,e'} \\ \hat{r}_{i,ee''} = \hat{r}_{i,e} - \hat{r}_{i,e''} \end{cases} \tag{6.3.6}$$

which respectively reflect user $u$'s interest in explanation $e$ over $e'$, and item $i$'s appropriateness for explanation $e$ over $e''$.

With the scores $\hat{r}_{u,ee'}$ and $\hat{r}_{u,ee''}$, we then adopt the BPR criterion [98] to minimize the following objective function:

$$\min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \sum_{e \in \mathcal{E}_{u,i}} \Big[ \sum_{e' \in \mathcal{E}/\mathcal{E}_u} - \ln \sigma(\hat{r}_{u,ee'}) + \sum_{e'' \in \mathcal{E}/\mathcal{E}_i} - \ln \sigma(\hat{r}_{i,ee''}) \Big] + \lambda \, ||\Theta||_F^2 \tag{6.3.7}$$

where $\sigma(\cdot)$ denotes the sigmoid function, $\mathcal{I}_u$ represents the set of items that user $u$ has interacted with, $\mathcal{E}_{u,i}$ is the set of explanations in the training set for the user-item pair $(u, i)$, $\mathcal{E}/\mathcal{E}_u$ and $\mathcal{E}/\mathcal{E}_i$ respectively correspond to explanations that user $u$ and item $i$ have not interacted with, $\Theta$ is the model parameter, and $\lambda$ is the regularization coefficient.

From Eq. (6.3.7), we can see that there are two explanation tasks to be learned respectively, corresponding to users and items. During the training stage, we allow them to be equally important, since we have a hyper-parameter $\mu$ in Eq. (6.3.5) to balance their importance during the testing stage. The effect of this parameter is studied in Section 6.6.1. After the model parameters are estimated, we rank explanations according to Eq. (6.2.2) for each user-item pair in the testing set. As we model the explanation ranking task under BPR criterion, we accordingly name our method Bayesian Personalized Explanation Ranking (BPER). To learn the model parameter $\Theta$, we draw on the widely used stochastic gradient descent algorithm to optimize the objective function in Eq. (6.3.7). Specifically, we first randomly initialize the parameters, and then repeatedly update them by uniformly taking

samples from the training set and computing the gradients w.r.t. the parameters, until the convergence of the algorithm. The complete learning steps are shown in Algorithm 1.

**Algorithm 1** Bayesian Personalized Explanation Ranking (BPER)

**Input:** training set $\mathcal{T}$, dimension of latent factors $d$, learning rate $\gamma$, regularization coefficient $\lambda$, iteration number $T$

**Output:** model parameter $\Theta = \{\mathbf{P}, \mathbf{Q}, \mathbf{O}^U, \mathbf{O}^I, \mathbf{b}^U, \mathbf{b}^I\}$

Initialize $\Theta$, including $\mathbf{P} \leftarrow \mathbb{R}^{|\mathcal{U}|\times d}$, $\mathbf{Q} \leftarrow \mathbb{R}^{|\mathcal{I}|\times d}$, $\mathbf{O}^U \leftarrow \mathbb{R}^{|\mathcal{E}|\times d}, \mathbf{O}^I \leftarrow \mathbb{R}^{|\mathcal{E}|\times d}, \mathbf{b}^U \leftarrow \mathbb{R}^{|\mathcal{E}|}, \mathbf{b}^I \leftarrow \mathbb{R}^{|\mathcal{E}|}$

**for** $t_1 = 1$ to $T$ **do**

  **for** $t_2 = 1$ to $|\mathcal{T}|$ **do**

    Uniformly draw $(u, i, e)$ from $\mathcal{T}$, $e'$ from $\mathcal{E}/\mathcal{E}_u$, and $e''$ from $\mathcal{E}/\mathcal{E}_i$

    $\hat{r}_{u,ee'} \leftarrow \hat{r}_{u,e} - \hat{r}_{u,e'}$, $\hat{r}_{i,ee''} \leftarrow \hat{r}_{i,e} - \hat{r}_{i,e''}$

    $x \leftarrow -\sigma(-\hat{r}_{u,ee'})$, $y \leftarrow -\sigma(-\hat{r}_{i,ee''})$

    $\mathbf{p}_u \leftarrow \mathbf{p}_u - \gamma \cdot (x \cdot (\mathbf{o}_e^U - \mathbf{o}_{e'}^U) + \lambda \cdot \mathbf{p}_u)$

    $\mathbf{q}_i \leftarrow \mathbf{q}_i - \gamma \cdot (y \cdot (\mathbf{o}_e^I - \mathbf{o}_{e''}^I) + \lambda \cdot \mathbf{q}_i)$

    $\mathbf{o}_e^U \leftarrow \mathbf{o}_e^U - \gamma \cdot (x \cdot \mathbf{p}_u + \lambda \cdot \mathbf{o}_e^U)$, $\mathbf{o}_{e'}^U \leftarrow \mathbf{o}_{e'}^U - \gamma \cdot (-x \cdot \mathbf{p}_u + \lambda \cdot \mathbf{o}_{e'}^U)$

    $\mathbf{o}_e^I \leftarrow \mathbf{o}_e^I - \gamma \cdot (y \cdot \mathbf{q}_i + \lambda \cdot \mathbf{o}_e^I)$, $\mathbf{o}_{e''}^I \leftarrow \mathbf{o}_{e''}^I - \gamma \cdot (-y \cdot \mathbf{q}_i + \lambda \cdot \mathbf{o}_{e''}^I)$

    $b_e^U \leftarrow b_e^U - \gamma \cdot (x + \lambda \cdot b_e^U)$, $b_{e'}^U \leftarrow b_{e'}^U - \gamma \cdot (-x + \lambda \cdot b_{e'}^U)$

    $b_e^I \leftarrow b_e^I - \gamma \cdot (y + \lambda \cdot b_e^I)$, $b_{e''}^I \leftarrow b_{e''}^I - \gamma \cdot (-y + \lambda \cdot b_{e''}^I)$

  **end for**

**end for**

### 6.3.3 BERT-enhanced BPER (BPER+)

The BPER model only exploits the IDs of users, items and explanations to infer their relation for explanation ranking. However, this makes the rich semantic features of the explanations, which could also capture the relation between explanations, under-explored. For example, "the acting is good" and "the acting is great" both convey a positive sentiment with a similar meaning, so their ranks are expected to

be close. Hence, we further investigate whether such features could help to enhance BPER. As a feature extractor, we opt for BERT [30], a well-known pre-trained language model, whose effectiveness has been demonstrated on a wide range of natural language understanding tasks. Specifically, we first add a special [CLS] token at the beginning of a textual explanation $e$, e.g., "[CLS] the acting is great". After passing it through BERT, we obtain the aggregate representation (corresponding to [CLS]) that encodes the explanation's overall semantics. To match the dimension of latent factors in our model, we apply a linear layer to this vector, resulting in $\mathbf{o}_e^{BERT}$. Then, we enhance the two ID-based explanation vectors $\mathbf{o}_e^U$ and $\mathbf{o}_e^I$ in Eq. (6.3.5) by multiplying $\mathbf{o}_e^{BERT}$, resulting in $\mathbf{o}_e^{U+}$ and $\mathbf{o}_e^{I+}$.

$$\begin{cases} \mathbf{o}_e^{U+} = \mathbf{o}_e^U \odot \mathbf{o}_e^{BERT} \\ \mathbf{o}_e^{I+} = \mathbf{o}_e^I \odot \mathbf{o}_e^{BERT} \end{cases} \tag{6.3.8}$$

To predict the score for $(u, i, e)$ triplet, we replace $\mathbf{o}_e^U$ and $\mathbf{o}_e^I$ in Eq. (6.3.5) with $\mathbf{o}_e^{U+}$ and $\mathbf{o}_e^{I+}$. Then we use Eq. (6.3.7) as the objective function, which can be optimized via back-propagation. In Eq. (6.3.8), we adopt the multiplication operation simply to verify the feasibility of incorporating semantic features. The model may be further improved by more sophisticated operations, e.g., multi-layer perceptron (MLP).

Notice that, BPER is a general method that only requires the IDs of users, items and explanations, which makes it very flexible when being adapted to other explanation styles (e.g., images [22]), whereas BPER+ is a domain-specific method that considers the semantic features extracted from textual explanations, so it could better perform ranking. As the first work on ranking explanations for recommendations, we opt to make both methods relatively simple for reproducibility purpose. In this way, it is also easy to observe the experimental results (such as the impact of explanation task on recommendation task), without the interference of other factors.

### 6.3.4 Relation between BPER, BPER+, CD, and PITF

In fact, our Bayesian Personalized Explanation Ranking (BPER) model is a type of Tensor Factorization (TF), so we analyze its relation to two closely related TF methods: Canonical Decomposition (CD) [12] and Pairwise Interaction Tensor Factorization (PITF) [99]. On one hand, in theory BPER can be considered as a special case of the CD model. Suppose the dimensionality of BPER is $2 \cdot d + 2$, we can reformulate it as CD in the following,

$$
\begin{aligned}
p_{u,k}^{CD} &= \begin{cases} \mu \cdot p_{u,k}, & \text{if } k \leq d \\ \mu, & \text{else} \end{cases} \\
q_{i,k}^{CD} &= \begin{cases} (1-\mu) \cdot q_{i,k}, & \text{if } k > d \text{ and } k \leq 2 \cdot d \\ 1-\mu, & \text{else} \end{cases} \\
o_{e,k}^{CD} &= \begin{cases} o_{e,k}^{U}, & \text{if } k \leq d \\ o_{e,k}^{I}, & \text{else if } k \leq 2 \cdot d \\ b_{e}^{U}, & \text{else if } k = 2 \cdot d + 1 \\ b_{e}^{I}, & \text{else} \end{cases}
\end{aligned}
\tag{6.3.9}
$$

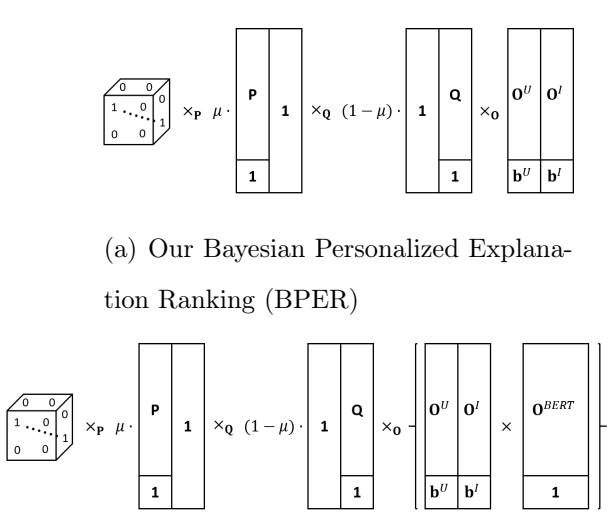
where the parameter $\mu$ is a constant.

On the other hand, PITF can be seen as a special case of our BPER. Formally, its predicted score $\hat{r}_{u,i,e}$ for the user-item-explanation triplet $(u, i, e)$ can be calculated by:

$$
\hat{r}_{u,i,e} = \mathbf{p}_u^\top \mathbf{o}_e^U + \mathbf{q}_i^\top \mathbf{o}_e^I = \sum_{k=1}^{d} p_{u,k} \cdot o_{e,k}^U + \sum_{k=1}^{d} q_{i,k} \cdot o_{e,k}^I \tag{6.3.10}
$$

We can see that our BPER degenerates to PITF if in Eq. (6.3.5) we remove the bias terms $b_e^U$ and $b_e^I$ and set the hyper-parameter $\mu$ to 0.5, which means that the two types of scores for users and items are equally important to the explanation ranking task.

Although CD is more general than our BPER, its performance may be affected by the data sparsity issue. Our BPER could mitigate this problem given its explicitly designed structure that may be difficult for CD to learn from scratch. When




(a) Our Bayesian Personalized Explanation Ranking (BPER)

(c) Canonical Decomposition (CD)

(b) Our BERT-enhanced BPER (BPER+)

(d) Pairwise Interaction Tensor Factorization (PITF)

Figure 6.2: Comparison of Tensor Factorization models. The three matrices (i.e., **P**, **Q**, **O**) are model parameters. Our BPER and BPER+ can be regarded as special cases of CD, while PITF can be seen as a special case of our BPER and BPER+.

comparing with PITF, we can find that the parameter $\mu$ in BPER is able to balance the importance of the two types of scores, corresponding to users and items, which makes our BPER more expressive than PITF and hence likely reach better ranking quality.

In a similar way, BPER+ can also be rewritten as CD or PITF. Concretely, by revising the last part of Eq. (6.3.9) as the following formula, BPER+ can be seen as CD. When $\mathbf{o}_e^{BERT} = [1, ..., 1]^\top$, BPER+ is equal to BPER, so it can be easily converted into PITF. The graphical illustration of the four models is shown in Fig. 6.2.

$$o_{e,k}^{CD} = \begin{cases} o_{e,k}^{U} \cdot o_{e,k}^{BERT}, & \text{if } k \leq d \\ o_{e,k}^{I} \cdot o_{e,k}^{BERT}, & \text{else if } k \leq 2 \cdot d \\ b_e^U, & \text{else if } k = 2 \cdot d + 1 \\ b_e^I, & \text{else} \end{cases} \tag{6.3.11}$$

### 6.3.5 Joint-Ranking on BPER (BPER-J)

Owing to BPER's flexibility to accommodate various explanation styles as discussed before, we perform the joint-ranking on it. Specifically, we incorporate the two tasks of explanation ranking and item recommendation into a unified multi-task learning framework, so as to find a good solution that benefits both of them.

For recommendation, we adopt Singular Value Decomposition (SVD) model [57] to predict the score $\hat{r}_{u,i}$ of user $u$ on item $i$:

$$\hat{r}_{u,i} = \mathbf{p}_u^\top \mathbf{q}_i + b_i = \sum_{k=1}^{d} p_{u,k} \cdot q_{i,k} + b_i \tag{6.3.12}$$

where $b_i$ is the bias term for item $i$. Notice that, the latent factors $\mathbf{p}_u$ and $\mathbf{q}_i$ are shared with those for explanation ranking in Eq. (6.3.5). In essence, item recommendation is also a ranking task that can be optimized using BPR criteria [98], so we first compute the preference difference $\hat{r}_{u,ii'}$ between a pair of items $i$ and $i'$ to a user $u$ as follows,

$$\hat{r}_{u,ii'} = \hat{r}_{u,i} - \hat{r}_{u,i'} \tag{6.3.13}$$

which can then be combined with the task of explanation ranking in Eq. (6.3.7) to form the following objective function for joint-ranking:

$$\begin{aligned} \min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \Big[ \sum_{i' \in \mathcal{I} / \mathcal{I}_u} - \ln \sigma(\hat{r}_{u,ii'}) + \alpha \sum_{e \in \mathcal{E}_{u,i}} \Big( \sum_{e' \in \mathcal{E} / \mathcal{E}_u} \\ - \ln \sigma(\hat{r}_{u,ee'}) + \sum_{e'' \in \mathcal{E} / \mathcal{E}_i} - \ln \sigma(\hat{r}_{i,ee''}) \Big) \Big] + \lambda \, ||\Theta||_F^2 \end{aligned} \tag{6.3.14}$$

where the parameter $\alpha$ can be fine-tuned to balance the learning of the two tasks.

We name this method **BPER-J** where J stands for joint-ranking. Similar to BPER, we can update each parameter of BPER-J via stochastic gradient descent (see Algorithm 2).

**Algorithm 2** Joint-Ranking on BPER (BPER-J)

**Input:** training set $\mathcal{T}$, dimension of latent factors $d$, learning rate $\gamma$, regularization coefficients $\alpha$ and $\lambda$, iteration number $T$

**Output:** model parameter $\Theta = \{\mathbf{P}, \mathbf{Q}, \mathbf{O}^U, \mathbf{O}^I, \mathbf{b}, \mathbf{b}^U, \mathbf{b}^I\}$

Initialize $\Theta$, including $\mathbf{P} \leftarrow \mathbb{R}^{|\mathcal{U}| \times d}$, $\mathbf{Q} \leftarrow \mathbb{R}^{|\mathcal{I}| \times d}$, $\mathbf{O}^U \leftarrow \mathbb{R}^{|\mathcal{E}| \times d}$, $\mathbf{O}^I \leftarrow \mathbb{R}^{|\mathcal{E}| \times d}$, $\mathbf{b} \leftarrow \mathbb{R}^{|\mathcal{I}|}$, $\mathbf{b}^U \leftarrow \mathbb{R}^{|\mathcal{E}|}$, $\mathbf{b}^I \leftarrow \mathbb{R}^{|\mathcal{E}|}$

**for** $t_1 = 1$ to $T$ **do**

  **for** $t_2 = 1$ to $|\mathcal{T}|$ **do**

    Uniformly draw $(u, i, e)$ from $\mathcal{T}$, $e'$ from $\mathcal{E}/\mathcal{E}_u$, $e''$ from $\mathcal{E}/\mathcal{E}_i$, and $i'$ from $\mathcal{I}/\mathcal{I}_u$

    $\hat{r}_{u,ee'} \leftarrow \hat{r}_{u,e} - \hat{r}_{u,e'}$, $\hat{r}_{i,ee''} \leftarrow \hat{r}_{i,e} - \hat{r}_{i,e''}$, $\hat{r}_{u,ii'} \leftarrow \hat{r}_{u,i} - \hat{r}_{u,i'}$

    $x \leftarrow -\alpha \cdot \sigma(-\hat{r}_{u,ee'})$, $y \leftarrow -\alpha \cdot \sigma(-\hat{r}_{i,ee''})$, $z \leftarrow -\sigma(-\hat{r}_{u,ii'})$

    $\mathbf{p}_u \leftarrow \mathbf{p}_u - \gamma \cdot (x \cdot (\mathbf{o}_e^U - \mathbf{o}_{e'}^U) + z \cdot (\mathbf{q}_i - \mathbf{q}_{i'}) + \lambda \cdot \mathbf{p}_u)$

    $\mathbf{q}_i \leftarrow \mathbf{q}_i - \gamma \cdot (y \cdot (\mathbf{o}_e^I - \mathbf{o}_{e''}^I) + z \cdot \mathbf{p}_u + \lambda \cdot \mathbf{q}_i)$

    $\mathbf{q}_{i'} \leftarrow \mathbf{q}_{i'} - \gamma \cdot (-z \cdot \mathbf{p}_u + \lambda \cdot \mathbf{q}_{i'})$

    $\mathbf{o}_e^U \leftarrow \mathbf{o}_e^U - \gamma \cdot (x \cdot \mathbf{p}_u + \lambda \cdot \mathbf{o}_e^U)$

    $\mathbf{o}_{e'}^U \leftarrow \mathbf{o}_{e'}^U - \gamma \cdot (-x \cdot \mathbf{p}_u + \lambda \cdot \mathbf{o}_{e'}^U)$

    $\mathbf{o}_e^I \leftarrow \mathbf{o}_e^I - \gamma \cdot (y \cdot \mathbf{q}_i + \lambda \cdot \mathbf{o}_e^I)$

    $\mathbf{o}_{e''}^I \leftarrow \mathbf{o}_{e''}^I - \gamma \cdot (-y \cdot \mathbf{q}_i + \lambda \cdot \mathbf{o}_{e''}^I)$

    $b_i \leftarrow b_i - \gamma \cdot (z + \lambda \cdot b_i)$

    $b_{i'} \leftarrow b_{i'} - \gamma \cdot (-z + \lambda \cdot b_{i'})$

    $b_e^U \leftarrow b_e^U - \gamma \cdot (x + \lambda \cdot b_e^U)$

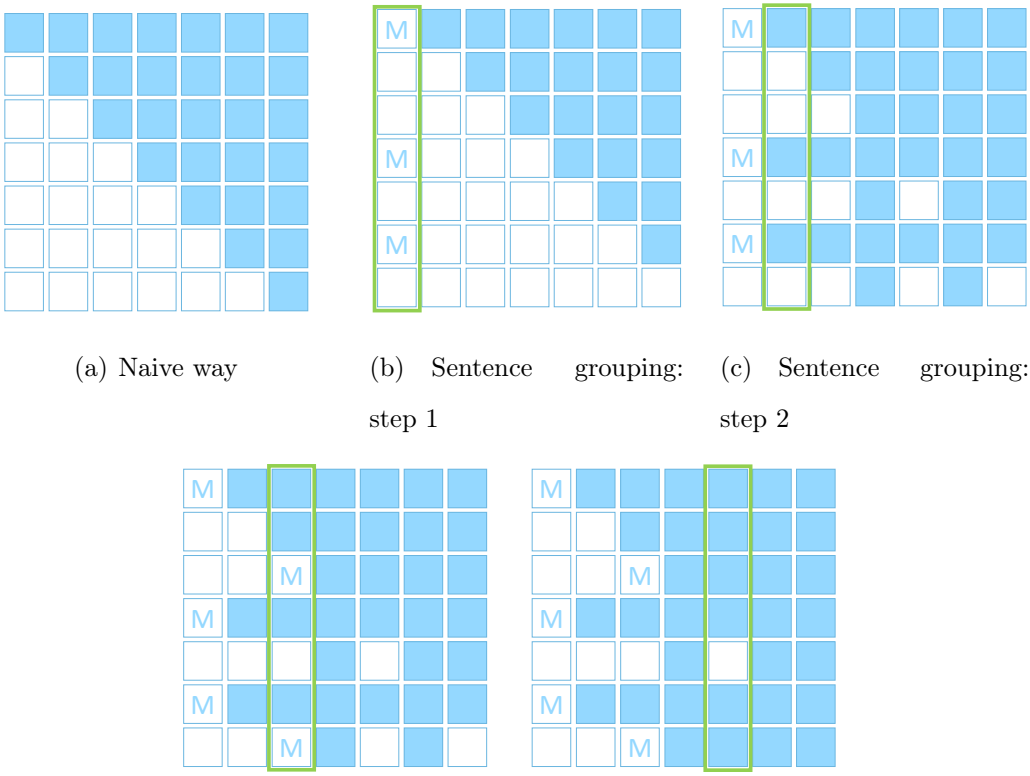
    $b_{e'}^U \leftarrow b_{e'}^U - \gamma \cdot (-x + \lambda \cdot b_{e'}^U)$

    $b_e^I \leftarrow b_e^I - \gamma \cdot (y + \lambda \cdot b_e^I)$

    $b_{e''}^I \leftarrow b_{e''}^I - \gamma \cdot (-y + \lambda \cdot b_{e''}^I)$

  **end for**

**end for**




(a) Naive way

(b) Sentence grouping: step 1

(c) Sentence grouping: step 2

(d) Sentence grouping: step 3

(e) Sentence grouping: step 4

Figure 6.3: Comparison between a naive way and our more efficient approach for sentence grouping. White cells denote similarity computation, while blue cells omit the computation. (a) shows the naive way to compute the similarity between any two sentences, which would take quadratic time. (b)-(e) show four example steps in our more efficient sentence grouping algorithm, where green rectangles denote query steps in LSH, and M denotes the matched duplicates.

## 6.4 Construction of User-Item-Explanation Interactions

For explanation ranking purpose, the datasets are expected to contain user-item-explanation interactions. We narrow down to the explanation sentences extracted from user reviews. The key problem lies in how to efficiently detect near-duplicates across different reviews, since it takes quadratic time to compute the similarity between any two sentences in a dataset. In the following, we present our approach

to finding duplicate sentences based on sentence grouping.

The advantage of sentence grouping is three-fold. First, it ensures the readability and expressiveness of the explanations, as they are extracted from real users' reviews based on the wisdom of the crowd. Second, it allows the explanations to be connected with both users and items, so that we can design collaborative filtering models to learn and predict such connections. Third, it makes explanation ranking and the automatic benchmark evaluation possible, since there are only a limited set of candidate explanations.

Computing the similarity between any two sentences in a dataset is computationally expensive. However, at each step of sentence grouping, it is actually unnecessary to compute the similarity for the already grouped sentences. Therefore, we can reduce the computation cost by removing those sentences (see Fig. 6.3 (b)-(e) for illustration). To find similar sentences more efficiently, we make use of Locality Sensitive Hashing (LSH) [96], which is able to conduct near-duplicate detection in sub-linear time. LSH consists of three major steps. First, a document (i.e., a sentence in our case) is converted into a set of $n$-shingles (a.k.a., $n$-grams). Second, the sets w.r.t. all documents are converted to short signatures via hashing, so as to reduce computation cost and meanwhile preserve document similarity. Third, the documents, whose similarity to a query document is greater than a pre-defined threshold, are returned. The detailed procedure of sentence grouping is shown in Algorithm 3.

Next, we discuss the implementation details. To make better use of all the available text in a dataset, for each record we concatenate the review text and the heading/tip. Then each piece of text is tokenized into sentences. In particular, a sentence is removed if it contains personal pronouns, e.g., "I" and "me", since explanations are expected to be more objective than subjective. We also calculate the frequency of nouns and adjectives in each sentence via NLTK[3], and only keep the sentences that contain both noun(s) and adjective(s), so as to obtain more informa-

[3]https://www.nltk.org

**Algorithm 3** Sentence Grouping via Locality-Sensitive Hashing (LSH)

**Input:** shingle size $n$, similarity threshold $t$, minimum group size $g$

**Output:** explanation set $\mathcal{E}$, groups of sentences $\mathcal{M}$

Pre-process textual data to obtain the sentence collection $\mathcal{S}$
$lsh \leftarrow MinHashLSH(t)$, $\mathcal{C} \leftarrow \varnothing$
**for** sentence $s$ in $\mathcal{S}$ **do**
    $m \leftarrow MinHash()$ // create MinHash for $s$
    **for** $n$-shingle $h$ in $s$ **do**
        $m.update(h)$ // convert $s$ into $m$ by encoding its $n$-shingles
    **end for**
    $lsh.insert(m)$, $\mathcal{C}.add(m)$ // $\mathcal{C}$: set of all sentences' MinHash
**end for**
$\mathcal{M} \leftarrow \varnothing$, $\mathcal{Q} \leftarrow \varnothing$ // $\mathcal{Q}$: set of queried sentences
**for** $m$ in $\mathcal{C}$ **do**
    **if** $m$ not in $\mathcal{Q}$ **then**
        $\mathcal{G} \leftarrow lsh.query(m)$ // $\mathcal{G}$: ID set of duplicate sentences
        **if** $\mathcal{G}.size > g$ **then**
            $\mathcal{M}.add(\mathcal{G})$ // only keep groups with enough sentences
            $\mathcal{E}.add(\mathcal{G}.get())$ // keep one explanation in each group
        **end if**
        **for** $m'$ in $\mathcal{G}$ **do**
            $lsh.remove(m')$, $\mathcal{Q}.add(m')$ // for efficiency
        **end for**
    **end if**
**end for**

tive explanations that evaluate certain item features. After the data pre-processing, we conduct sentence grouping via an open-source LSH [96] package Datasketch[4]. Notice that, we apply it to all sentences in a dataset, rather than that of a particular item, because it is easier to find common expressions from a large amount of sentences. When creating MinHash for each sentence, we set the shingle size $n$ to 2 so as to preserve the word ordering and meanwhile distinguish positive sentiment from negative sentiment (e.g., "is good" v.s. "not good"). We test the similarity threshold $t$ of querying sentences from [0.5, 0.6, ..., 0.9], and find that the results with 0.9 are the best.

## 6.5 Experimental Setup

### 6.5.1 Datasets

We construct our datasets on three domains: Amazon Movies & TV[5] (movie), TripAdvisor[6] (hotel) and Yelp[7] (restaurant). In each of the datasets, a record is comprised of user ID, item ID, overall rating in the scale of 1 to 5, and textual review. After splitting reviews into sentences, we apply sentence grouping (in Algorithm 3) over them to obtain a large amount of sentence groups. A group is removed if its number of sentences is smaller than 5 so as to retain commonly seen explanations. We then assign each of the remaining groups an ID that we call an explanation ID. Eventually, a user-item pair may be connected to none, one or multiple explanation IDs since the review of the user-item pair may contain none, one or multiple explanation sentences. We remove the user-item pairs that are not connected to any explanation ID, and the remaining records are thus user-item-explanation triplets. The statistics of the three datasets are presented in Table 6.2. As it can be seen, the data sparsity issue on the three datasets is very severe.

[4]http://ekzhu.com/datasketch/lsh.html

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

[6]https://www.tripadvisor.com

[7]https://www.yelp.com/dataset/challenge

Table 6.2: Statistics of the datasets. Density is #triplets divided by #users × #items × #explanations.

| | Amazon | TripAdvisor | Yelp |
|---|---|---|---|
| # of users | 109,121 | 123,374 | 895,729 |
| # of items | 47,113 | 200,475 | 164,779 |
| # of explanations | 33,767 | 76,293 | 126,696 |
| # of $(u, i)$ pairs | 569,838 | 1,377,605 | 2,608,860 |
| # of $(u, i, e)$ triplets | 793,481 | 2,618,340 | 3,875,118 |
| # of explanations / $(u, i)$ pair | 1.39 | 1.90 | 1.49 |
| Density ($\times 10^{-10}$) | 45.71 | 13.88 | 2.07 |

Table 6.3 shows 5 example explanations taken from the three datasets. As we can see, all the explanations are quite concise and informative, not only because LSH favors short text, but also because people tend to express their opinions using common and concise phrases. This could prevent from overwhelming users, a critical issue for explainable recommendation [46]. Also, short explanations can be mobile-friendly, since it is difficult for a small screen to fit much content. Moreover, the explanations from different datasets well suit the target application domains, such as "a wonderful movie for all ages" for movies and "comfortable hotel with good facilities" for hotels. Explanations with negative sentiment can also be observed, e.g., "the place is awful", which can be used to justify why some items are dis-recommended [133].

### 6.5.2 Compared Methods

To evaluate the performance of explanation ranking task, where the user-item pairs are given, we adopt the following baselines. Notice that, we omit the comparison with Tucker Decomposition (TD) [113], because it takes cubic time to run and we also find that it does not perform better than CD in our trial experiment.

Table 6.3: Example explanations on the three datasets. Occurrence denotes the number of records that contain the explanation.

| **Explanation** | **Occurrence** |
|---|---|
| **Amazon Movies & TV** | |
| Great story | 3307 |
| Don't waste your money | 834 |
| The acting is great | 760 |
| The sound is okay | 11 |
| A wonderful movie for all ages | 6 |
| **TripAdvisor** | |
| Great location | 61993 |
| The room was clean | 6622 |
| The staff were friendly and helpful | 2184 |
| Bad service | 670 |
| Comfortable hotel with good facilities | 8 |
| **Yelp** | |
| Great service | 46413 |
| Everything was delicious | 5237 |
| Prices are reasonable | 2914 |
| This place is awful | 970 |
| The place was clean and the food was good | 6 |

- **RAND**: It is a weak baseline that randomly picks up explanations from the explanation collection $\mathcal{E}$. It is devised to examine whether personalization is needed for explanation ranking.

- **RUCF**: Revised User-based Collaborative Filtering. Because traditional CF methods [100, 103] cannot be directly applied to the ternary data, we make some modifications to their formula, following [50]. The similarity between two users is measured by their associated explanation sets via Jaccard Index. When predicting a score for the $(u, i, e)$ triplet, we first find users associated with the same item $i$ and explanation $e$, i.e., $\mathcal{U}_i \cap \mathcal{U}_e$, from which we then find the ones appearing in user $u$'s neighbor set $\mathcal{N}_u$.

$$\hat{r}_{u,i,e} = \sum_{u' \in \mathcal{N}_u \cap (\mathcal{U}_i \cap \mathcal{U}_e)} s_{u,u'} \text{ where } s_{u,u'} = \frac{|\mathcal{E}_u \cap \mathcal{E}_{u'}|}{|\mathcal{E}_u \cup \mathcal{E}_{u'}|} \tag{6.5.15}$$

- **RICF**: Revised Item-based Collaborative Filtering. This method predicts a score for a triplet from the perspective of items, whose formula is similar to Eq. (6.5.15).

- **CD**: Canonical Decomposition [12] as shown in Eq. (6.3.4). This method only predicts one score instead of two for the triplet $(u, i, e)$, so its objective function shown below is slightly different from ours in Eq. (6.3.7).

$$\min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \sum_{e \in \mathcal{E}_{u,i}} \sum_{e' \in \mathcal{E} / \mathcal{E}_{u,i}} - \ln \sigma(\hat{r}_{u,i,ee'}) + \lambda \, ||\Theta||_F^2 \tag{6.5.16}$$

  where $\hat{r}_{u,i,ee'} = \hat{r}_{u,i,e} - \hat{r}_{u,i,e'}$ is the score difference between a pair of explanations.

- **PITF**: Pairwise Interaction Tensor Factorization [99]. It makes prediction for a triplet based on Eq. (6.3.10), and its objective function is identical to CD's in Eq. (6.5.16).

To verify the effectiveness of the joint-ranking framework, in addition to our method BPER-J, we also present the results of two baselines: CD [12] and PITF [99]. Since CD and PITF are not originally designed to accomplish the two tasks

of item ranking and explanation ranking together, we first allow them to make prediction for a user-item pair $(u, i)$ via the inner product of the associated latent factors, i.e., $\hat{r}_{u,i} = \mathbf{p}_u^T \mathbf{q}_i$, and then combine this task with explanation ranking in a multi-task learning framework whose objective function is given below:

$$\min_{\Theta} \sum_{u \in \mathcal{U}} \sum_{i \in \mathcal{I}_u} \Big[ \sum_{i' \in \mathcal{I}/\mathcal{I}_u} - \ln \sigma(\hat{r}_{u,ii'}) + \alpha \sum_{e \in \mathcal{E}_{u,i}} \sum_{e' \in \mathcal{E}/\mathcal{E}_{u,i}} - \ln \sigma(\hat{r}_{u,i,ee'}) \Big] + \lambda \, ||\Theta||_F^2 \quad (6.5.17)$$

where $\hat{r}_{u,ii'} = \hat{r}_{u,i} - \hat{r}_{u,i'}$ is the difference between a pair of items. We name them CD-J and PITF-J respectively, where J denotes joint-ranking.

### 6.5.3 Implementation Details

To evaluate the performance of both recommendation and explanation, we adopt four commonly used ranking-oriented metrics in recommender systems: Normalized Discounted Cumulative Gain (**NDCG**), Precision (**Pre**), Recall (**Rec**) and **F1**. We evaluate on top-10 ranking for both recommendation and explanation tasks.

We randomly divide each dataset into training (70%) and testing (30%) sets, and guarantee that each user/item/explanation has at least one record in the training set. The splitting process is repeated for 5 times. For validation, we randomly draw 10% records from training set. After hyper-parameters tuning, the average performance on the 5 testing sets is reported.

We implemented all the methods in Python. For TF-based methods, including CD, PITF, CD-J, PITF-J, and our BPER and BPER-J, we search the dimension of latent factors $d$ from [10, 20, 30, 40, 50], regularization coefficient $\lambda$ from [0.001, 0.01, 0.1], learning rate $\gamma$ from [0.001, 0.01, 0.1], and maximum iteration number $T$ from [100, 500, 1000]. As to joint-ranking of CD-J, PITF-J and our BPER-J, the regularization coefficient $\alpha$ on explanation task is searched from [0, 0.1, ..., 0.9, 1]. For the evaluation of joint-ranking, we first evaluate the performance of item recommendation for users, followed by the evaluation of explanation ranking on those correctly predicted user-item pairs. For our methods BPER and BPER-J, the parameter $\mu$ that balances user and item scores for explanation ranking is searched

from [0, 0.1, ..., 0.9, 1]. After parameter tuning, we use $d = 20$, $\lambda = 0.01$, $\gamma = 0.01$ and $T = 500$ for our methods, while the other parameters $\alpha$ and $\mu$ are dependent on the datasets.

The configuration of BPER+ is slightly different, owing to the textual content of the explanations. We adopted the pre-trained BERT from huggingface[8], and implemented the model in Python with PyTorch[9]. We set batch size to 128, $d = 20$ and $T = 5$. After parameter tuning, we set learning rate $\gamma$ to 0.0001 on Amazon, and 0.00001 on both TripAdvisor and Yelp.

## 6.6 Results and Analysis

In this section, we first present the comparison of our methods BPER and BPER+ with baselines regarding explanation ranking. Then, we study the capability of our methods in dealing with varying data sparseness. Third, we show a case study of explanation ranking for both recommendation and disrecommendation. Lastly, we analyze the joint-ranking results of three TF-based methods.

### 6.6.1 Comparison of Explanation Ranking

Experimental results for explanation ranking on the three datasets are shown in Table 6.4. We see that each method's performance on the four metrics (i.e., NDCG, Precision, Recall and F1) are fairly consistent across the three datasets. The method RAND is among the weakest baselines, because it randomly selects explanations without considering user and item information, which implies that the explanation ranking task is non-trivial. CD performs even worse than RAND, because of the sparsity issue in the ternary data (see Table 6.2), for which CD may not be able to mitigate as discussed in Section 6.3.2. CF-based methods, i.e., RUCF and RICF, largely advance the performance of RAND, as they take into account the information of either users or items, which confirms the important role of personalization for

[8]https://huggingface.co/bert-base-uncased

[9]https://pytorch.org

Table 6.4: Performance comparison of all methods on the top-10 explanation ranking in terms of NDCG, Precision (Pre), Recall (Rec) and F1 (%). The best performing values are boldfaced, and the second best underlined. Improvements are made by BPER+ over the best baseline PITF (* indicates the statistical significance over PITF for $p < 0.01$ via Student's t-test).

| | Amazon | | | | TripAdvisor | | | | Yelp | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | NDCG@10 | Pre@10 | Rec@10 | F1@10 | NDCG@10 | Pre@10 | Rec@10 | F1@10 | NDCG@10 | Pre@10 | Rec@10 | F1@10 |
| CD | 0.001 | 0.001 | 0.007 | 0.002 | 0.001 | 0.001 | 0.003 | 0.001 | 0.000 | 0.000 | 0.003 | 0.001 |
| RAND | 0.004 | 0.004 | 0.027 | 0.006 | 0.002 | 0.002 | 0.011 | 0.004 | 0.001 | 0.001 | 0.007 | 0.002 |
| RUCF | 0.341 | 0.170 | 1.455 | 0.301 | 0.260 | 0.151 | 0.779 | 0.242 | 0.040 | 0.020 | 0.125 | 0.033 |
| RICF | 0.417 | 0.259 | 1.797 | 0.433 | 0.031 | 0.020 | 0.087 | 0.030 | 0.037 | 0.026 | 0.137 | 0.042 |
| PITF | 2.352 | 1.824 | 14.125 | 3.149 | 1.239 | 1.111 | 5.851 | 1.788 | 0.712 | 0.635 | 4.172 | 1.068 |
| **BPER** | 2.630* | **1.942*** | **15.147*** | **3.360*** | 1.389* | 1.236* | 6.549* | 1.992* | 0.814* | 0.723* | **4.768*** | 1.218* |
| **BPER+** | **2.877*** | 1.919* | 14.936* | 3.317* | **2.096*** | **1.565*** | **8.151*** | **2.515*** | **0.903*** | **0.731*** | 4.544* | **1.220*** |
| Improvement (%) | 22.352 | 5.229 | 5.739 | 5.343 | 69.073 | 40.862 | 39.314 | 40.665 | 26.861 | 15.230 | 8.925 | 14.228 |

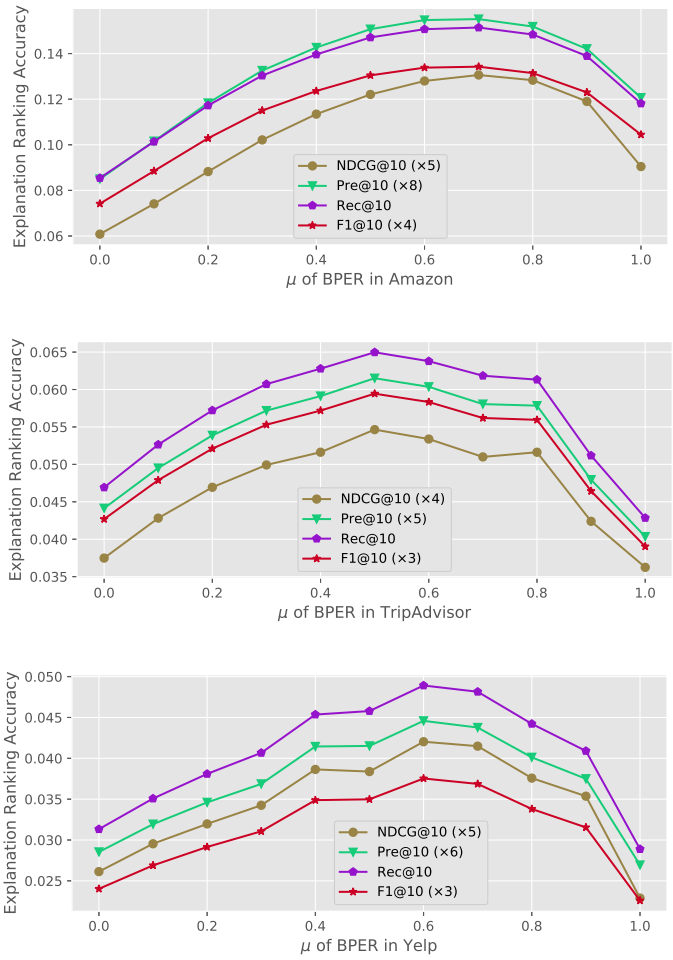



Figure 6.4: The effect of $\mu$ in BPER on explanation ranking in three datasets. NDCG@10, Pre@10 and F1@10 are linearly scaled for better visualization.

explanation ranking. However, their performance is still limited due to data sparsity. PITF and our BPER/BPER+ outperform the CF-based methods by a large margin, as they not only address the data sparsity issue via their MF-like model structure, but also take each user's and item's information into account via latent factors. Most importantly, our method BPER significantly outperforms the strongest baseline PITF, owing to its ability to produce two sets of scores, corresponding to users and items respectively, and the parameter $\mu$ that can balance their relative importance to the explanation ranking task. Lastly, BPER+ further improves BPER on most of the metrics across the three datasets, especially on NDCG that cares about the ranking order, which can be attributed to the consideration of the explanations' semantic features as well as BERT's strong language modeling capability.

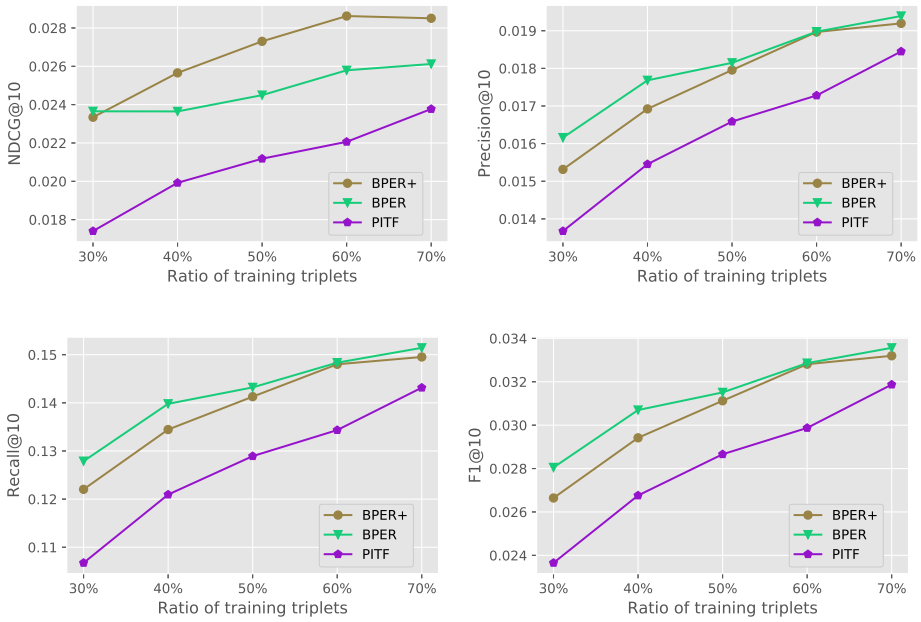



Figure 6.5: Ranking performance of three TF-based methods w.r.t. varying sparseness of training data on Amazon dataset.

Next, we further analyze the parameter $\mu$ of BPER that controls the contributions of user scores and item scores in Eq. (6.3.5). As it can be seen in Fig. 6.4, the curves of NDCG, Precision, Recall and F1 are all bell-shaped, where the performance improves significantly with the increase of $\mu$ until it reaches an optimal point, and then it drops sharply. Due to the characteristics of different application domains, the optimal points vary among the three datasets, i.e., 0.7 for both Amazon and Yelp and 0.5 for TripAdvisor. We omit the figures of BPER+, because the pattern is similar.

### 6.6.2 Results on Varying Data Sparseness

As discussed earlier, the sparsity issue of user-item-explanation triple-wise data is severer than that of traditional user-item pair-wise data. To investigate how different methods deal with varying spareness, we further divide the Amazon dataset into different splits, where the ratio of the training triplets to the whole dataset ranges from 30% to 70%. For comparison with our BPER and BPER+, we include the most competitive baseline PITF. Fig. 6.5 shows the ranking performance of the

Table 6.5: Top-5 explanations selected by BPER and PITF for two given user-item pairs, corresponding to recommendation and disrecommendation, on Amazon Movies & TV dataset. The ground-truth explanations are unordered. Matched explanations are emphasized in italic font.

| **Ground-truth** | **BPER** | **PITF** |
|---|---|---|
| Special effects | *Special effects* | Great special effects |
| Great story | Good acting | Great visuals |
| Wonderful movie | This is a great movie | Great effects |
| | *Great story* | *Special effects* |
| | Great special effects | Good movie |
| The acting is terrible | *The acting is terrible* | Good action movie |
| | The acting is bad | Low budget |
| | The acting was horrible | Nothing special |
| | It's not funny | The acting is poor |
| | Bad dialogue | The acting is bad |

three methods w.r.t. varying spareness. The ranking results are quite consistent on the four metrics (i.e., NDCG, Precision, Recall and F1). Moreover, with the increase of the amount of training triplets, the performance of all the three methods goes up proportionally. Particularly, the performance gap between our BPER/BPER+ and PITF is quite large, especially when the ratio of training data is small (e.g., 30%). These observations demonstrate our methods' better capability in mitigating data sparsity issue, and also prove the rationale of our solution which converts triplets into two groups of binary relation.

### 6.6.3 Case Study of Explanation Ranking

To better understand how explanation ranking works, we present a case study comparing our method BPER and the most effective baseline PITF on Amazon Movies

& TV dataset in Table 6.5. The two cases in the table respectively correspond to recommendation and disrecommendation. In the first case (i.e., recommendation), there are three ground-truth explanations, praising the movie's "special effects", "story" and overall quality. Generally speaking, the top-5 explanations resulting from both BPER and PITF are positive, and relevant to the ground-truth, because the two methods are both effective in terms of explanation ranking. However, since PITF's ranking ability is relatively weaker than our BPER, its explanations miss the key feature "story" that the user also cares about.

In the second case (i.e., disrecommendation), the ground-truth explanation is a negative comment about the target movie's "acting". Although the top explanations made by both BPER and PITF contain negative opinions with regard to this aspect, their ranking positions are quite different (i.e., top-3 for our BPER vs. bottom-2 for PITF). Moreover, we notice that for this disrecommendation, PITF places a positive explanation in the 1st position, i.e., "good action movie", which not only contradicts the other two explanations, i.e., "the acting is poor/bad", but also mismatches the disrecommendation goal. Again, this showcases our model's effectiveness for explanation ranking.

### 6.6.4 Effect of Joint-Ranking

We perform joint-ranking for three TF-based models, i.e., BPER-J, CD-J and PITF-J. Because of the consistency between different datasets, we only show the experimental results on Amazon and TripAdvisor. In Fig. 6.6, we study the effect of the parameter $\alpha$ to both explanation ranking and item ranking in terms of F1 (results on the other three metrics are consistent). In each sub-figure, the green dotted line represents the performance of explanation ranking task without joint-ranking, whose value is taken from Table 6.4. As we can see, all the points on the explanation curve (in red) are above this line when $\alpha$ is greater than 0, suggesting that the explanation task benefits from the recommendation task under the joint-ranking framework. In particular, the explanation performance of CD-J improves dramatically under the

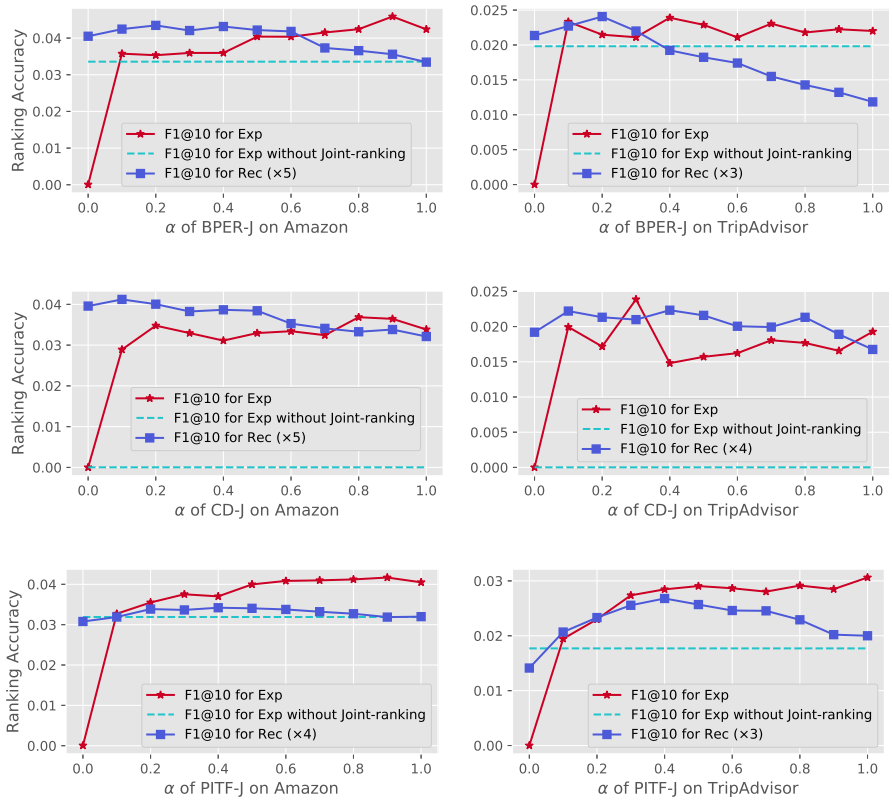



Figure 6.6: The effect of $\alpha$ in three TF-based methods with joint-ranking on two datasets. Exp and Rec respectively denote the Explanation and Recommendation tasks. F1@10 for Rec is linearly scaled for better visualization.

joint-ranking framework, since its recommendation task suffers less from the data sparsity issue than the explanation task as discussed in Section 6.3.2. It in turn helps to better rank the explanations. Meanwhile, for the recommendation task, all the three models degenerate to BPR when $\alpha$ is set to 0. Therefore, on the recommendation curves (in blue), any points, whose values are greater than that of the starting point, gain profits from the explanation task as well. All these observations show the effectiveness of our joint-ranking framework in terms of enabling the two tasks to benefit from each other.

In Table 6.6, we make a self-comparison of the three methods in terms of NDCG and F1 (the other two metrics are similar). In this table, "Non-joint-ranking" corresponds to each model's performance with regard to explanation or recommendation when the two tasks are individually learned. In other words, the explanation perfor-

Table 6.6: Self-comparison of three TF-based methods on two datasets with and without joint-ranking in terms of NDCG (N for short) and F1 (denoted as F). Top-10 results are evaluated for both explanation (Exp) and recommendation (Rec) tasks. The improvements are made by the best performance of each task under joint-ranking over that without it (i.e., in this case the two tasks are separately learned).

| | | Amazon | | | | TripAdvisor | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Exp (%) | | Rec (‰) | | Exp (%) | | Rec (‰) | |
| | | N | F | N | F | N | F | N | F |
| | | **BPER-J** | | | | | | | |
| Non-joint-ranking | | 2.6 | 3.4 | 6.6 | 8.1 | 1.4 | 2.0 | 5.3 | 7.1 |
| Joint-ranking | Best Exp | 3.3 ↑ | 4.6 ↑ | 5.7 ↓ | 7.1 ↓ | 1.6 ↑ | 2.4 ↑ | 5.0 ↓ | 6.4 ↓ |
| | Best Rec | 2.6 ↕ | 3.5 ↑ | 7.1 ↑ | 8.7 ↑ | 1.5 ↑ | 2.1 ↑ | 6.3 ↑ | 8.0 ↑ |
| Improvement (%) | | 26.9 | 35.3 | 7.6 | 7.4 | 14.3 | 20.0 | 18.9 | 11.3 |
| | | **CD-J** | | | | | | | |
| Non-joint-ranking | | 0.0 | 0.0 | 6.5 | 7.9 | 0.0 | 0.0 | 4.5 | 4.8 |
| Joint-ranking | Best Exp | 2.6 ↑ | 3.7 ↑ | 5.5 ↓ | 6.7 ↓ | 1.7 ↑ | 2.4 ↑ | 4.6 ↑ | 5.2 ↑ |
| | Best Rec | 1.9 ↑ | 2.9 ↑ | 6.8 ↑ | 8.2 ↑ | 9.6 ↑ | 1.5 ↑ | 4.9 ↑ | 5.6 ↑ |
| Improvement (%) | | Inf | Inf | 4.6 | 3.8 | Inf | Inf | 8.9 | 16.7 |
| | | **PITF-J** | | | | | | | |
| Non-joint-ranking | | 2.4 | 3.2 | 6.5 | 7.7 | 1.2 | 1.8 | 4.3 | 4.7 |
| Joint-ranking | Best Exp | 3.0 ↑ | 4.2 ↑ | 6.4 ↓ | 8.0 ↑ | 2.0 ↑ | 2.9 ↑ | 6.0 ↑ | 7.6 ↑ |
| | Best Rec | 2.8 ↑ | 3.7 ↑ | 7.1 ↑ | 8.5 ↑ | 2.0 ↑ | 2.8 ↑ | 7.0 ↑ | 8.9 ↑ |
| Improvement (%) | | 25.0 | 31.3 | 9.2 | 10.4 | 66.7 | 61.1 | 62.8 | 89.4 |

mance is taken from Table 6.4, and the recommendation performance is evaluated when $\alpha = 0$. "Best Exp" and "Best Rec" denote the best performance of each method on respectively explanation task and recommendation task under the joint-ranking framework. As we can see, when the recommendation performance is the best for all the models with joint-ranking, the explanation performance is always improved. Although minor recommendation accuracy is sacrificed when the explanation task reaches the best performance, we can always find points where both of the two tasks are improved, e.g., on the top left of Fig. 6.6 when $\alpha$ is in the range of 0.1 to 0.6 for BPER-J on Amazon. This again demonstrates our joint-ranking framework's capability in finding good solutions for both tasks.

## 6.7 Summary

In this chapter, we formulate the recommendation explanation problem as a ranking task, with an attempt to achieve standard offline evaluation of explainability. To facilitate the development of explainable recommendation, we construct three large datasets, based on which we develop two effective models to address the data sparsity issue. With the quantitative measure of explainability as well as the effective models, we design an item-explanation joint-ranking framework that can improve the performance of both recommendation and explanation tasks. Besides improving the recommendation performance by providing explanations, the joint-ranking framework could be potentially extended to other objectives, such as recommendation serendipity [21] and fairness [108].

# Chapter 7

# Conclusion and Future Work

This thesis introduces datasets, approaches and metrics for explainable recommendation. We first conclude the chapters that correspond to these contributions, and then discuss open problems in this field as well as a broader scope of XAI.

## 7.1 Conclusion

In Chapter 3, we propose CAESAR (Context-Aware Explanation based on Supervised Attention for Recommendations) that can model explicit contextual features via supervised attention mechanism to make the selected features match to a user's preference. We also design a two-level attention mechanism, i.e., feature-level and context-level, to adaptively distinguish the importance of different contexts and their related contextual features to both recommendation and explanation. Experimental results via human evaluation suggest that the context-aware explanations are indeed more useful than context-unaware explanations. Moreover, context-aware explanation could be incorporated into conversational recommendation scenario [132, 25], where recommendations can be better explained in accordance with a user's mobile environment.

In Chapter 4, we present a NEural TEmplate (NETE) generation approach to improve explanation quality and flexibility simultaneously. It unifies the merits of both natural language generation models and pre-defined templates. Experiments on

real-world datasets provide substantial evidence that NETE is capable of producing diverse, high-quality, and controllable natural language explanations. Furthermore, our human evaluation confirms that the explanations from our model are perceived helpful by users. This approach can be very useful in practice. For instance, when a user proactively asks the system to explain a particular feature of a recommendation, e.g., "price", we would expect the model to generate a more targeted explanation. Moreover, our proposed four new metrics for explainable recommendation could also benefit other fields, such as dialogue systems [128, 135] which sometimes return universal reply, e.g., "I do not know". Then, we can adopt the sentence diversity metric to quantitatively measure this issue.

In Chapter 5, we design a simple and effective learning objective to address the problem that the well-known Transformer model could not make use of user and item IDs for recommendation explanation generation. We evaluate the explanations generated by our PETER (PErsonalized Transformer for Explainable Recommendation) on not only text quality metrics, but also metrics that focus on explainability from the angle of item features. Extensive experiments show that PETER can outperform state-of-the-art baselines on large datasets. Our solution may shed light on a broader scope of fields that also need personalization, e.g., personalized conversational systems. In addition, it may point out a way for Transformer to deal with heterogeneous inputs, e.g., text and images in multi-modal AI.

In Chapter 6, we construct three benchmark datasets for explanation ranking, on which explainability can be evaluated quantitatively, just as recommendation task. This would enable standard offline evaluation of explainable recommendation, and also encourage researchers to develop more effective models. To mitigate the sparsity issue in the user-item-explanation data, we propose to perform two sets of matrix factorization, and apply this idea to two types of models. Based on the data and models, we further propose an item-explanation joint-ranking framework to improve the recommendation performance by explanation, and study the relationship of the two tasks.

## 7.2 Future Work

In Chapter 6, we show that the recommendation performance can be improved by adjusting the explanation task in the joint-ranking formulation. What behind the improved recommendation accuracy is that some particular explanations may be selected purposely. A problem follows up: are these explanations faithful to the recommendations? What if they are chosen by the model simply because they can cheat and manipulate users to achieve the goal of improving clicking rates? This would be unethical, and also make the recommender system untrustworthy once discovered. Hence, further measures should be taken to address this type of unintentionally negative and harmful effect [33].

Then, we discuss the potential bias issue in explainable recommendation. In this thesis, we design two natural language explanation generation approaches based on respectively RNN (Chapter 4) and Transformer (Chapter 5). More recently, we develop a more effective version [68] based on the pre-trained language model GPT-2 [95] (not included in this thesis). For the former, a number of RNN-based explanation approaches [81, 109] employ Word2Vec [86, 87] to initialize the word embeddings, but gender bias is found in Word2Vec [9]. For example, such an association can be obtained via vector operation: "man" - "doctor" = "woman" - "nurse". For the latter, i.e., pre-trained models, recent studies [106, 74] report that they exhibit societal bias towards certain groups of people. For example, given the prefix "the man (or woman) performing surgery is a", the model may produce "doctor (or nurse)" [74]. Although bias or stereotype does exist in real world, machines are expected to be neutral, inclusive, fair and unbiased, otherwise this type of issue could be amplified by their wide deployment. As a response, we should take measures to mitigate bias and to achieve fairness.

There are some works [72, 73] that study fairness in recommender systems, but bias and fairness in explainable recommendation have received relatively less attention. In this regard, we have the following concerns. Does the bias issue as observed in other applications (e.g., conversational systems [106]) also exist in explanation

generation models? Further, could the generated explanations possess bias against people who have different demographic attributes, such as gender, age, income, religion and race? In what form does bias exist in pre-trained models? This then leads back to the key problem of interpretability in XAI. Since pre-trained models have been successfully adapted to a variety of AI applications, such as search engines [140] and conversational systems [130], addressing the bias issue in one application would shed light on the others. More importantly, reducing bias would also promote fairness, benefit many people, and even make the world a better place.

# Bibliography

[1] Gregory D Abowd, Anind K Dey, Peter J Brown, Nigel Davies, Mark Smith, and Pete Steggles. Towards a better understanding of context and context-awareness. In *International symposium on handheld and ubiquitous computing*, pages 304–307. Springer, 1999.

[2] Gediminas Adomavicius and Alexander Tuzhilin. Context-aware recommender systems. In Bracha Shapira, editor, *Recommender Systems Handbook*, chapter 6, pages 191–226. Springer, 2 edition, 2015.

[3] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. Learning heterogeneous knowledge base embeddings for explainable recommendation. *Algorithms*, 11(9):137, 2018.

[4] John Arevalo, Thamar Solorio, Manuel Montes-y Gómez, and Fabio A González. Gated multimodal units for information fusion. In *ICLR Workshop*, 2017.

[5] Krisztian Balog and Filip Radlinski. Measuring recommendation explanation quality: The conflicting goals of explanations. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 329–338, 2020.

[6] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the fifth ACM conference on Recommender systems*, pages 301–304, 2011.

[7] Ramesh Baral, XiaoLong Zhu, SS Iyengar, and Tao Li. Reel: Review aware explanation of location recommendation. In *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization*, pages 23–32. ACM, 2018.

[8] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166, 1994.

[9] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *Advances in neural information processing systems*, 2016.

[10] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *Advances in neural information processing systems*, 2020.

[11] Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161, 2018.

[12] J Douglas Carroll and Jih-Jie Chang. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

[13] Rose Catherine and William Cohen. Transnets: Learning to transform for recommendation. In *Proceedings of the eleventh ACM conference on recommender systems*, pages 288–296, 2017.

[14] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. Neural attentional rating regression with review-level explanations. In *Proceedings of the 2018 World Wide Web Conference*, pages 1583–1592, 2018.

[15] Guanliang Chen and Li Chen. Augmenting service recommender systems by incorporating contextual opinions from user reviews. *User Modeling and User-Adapted Interaction*, 25(3):295–329, 2015.

[16] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. Generate natural language explanations for recommendation. In *Proceedings of SIGIR'19 Workshop on ExplainAble Recommendation and Search*. ACM, 2019.

[17] Hanxiong Chen, Shaoyun Shi, Yunqi Li, and Yongfeng Zhang. Neural collaborative reasoning. In *Proceedings of The Web Conference 2021*, 2021.

[18] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. Attentive collaborative filtering: Multimedia recommendation with item-and component-level attention. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 335–344. ACM, 2017.

[19] Li Chen and Feng Wang. Explaining recommendations based on feature sentiments in product reviews. In *Proceedings of the 22nd international conference on intelligent user interfaces*, pages 17–28, 2017.

[20] Li Chen, Dongning Yan, and Feng Wang. User evaluations on sentiment-based recommendation explanations. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 9(4):1–38, 2019.

[21] Li Chen, Yonghua Yang, Ningxia Wang, Keping Yang, and Quan Yuan. How serendipity improves user satisfaction with recommendations? a large-scale user evaluation. In *The World Wide Web Conference*, pages 240–250, 2019.

[22] Xu Chen, Hanxiong Chen, Hongteng Xu, Yongfeng Zhang, Yixin Cao, Zheng Qin, and Hongyuan Zha. Personalized fashion recommendation with visual explanations based on multimodal attention network: Towards visually explainable recommendation. In *Proceedings of the 42nd International ACM*

*SIGIR Conference on Research and Development in Information Retrieval*, pages 765–774, 2019.

[23] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. Learning to rank features for recommendation over multiple categories. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 305–314, 2016.

[24] Xu Chen, Yongfeng Zhang, and Zheng Qin. Dynamic explainable recommendation based on neural attentive models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 53–60, 2019.

[25] Zhongxia Chen, Xiting Wang, Xing Xie, Mehul Parsana, Akshay Soni, Xiang Ao, and Enhong Chen. Towards explainable conversational recommendation. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, 2020.

[26] Zhongxia Chen, Xiting Wang, Xing Xie, Tong Wu, Guoqing Bu, Yining Wang, and Enhong Chen. Co-attentive multi-task learning for explainable recommendation. In *IJCAI*, pages 2137–2143, 2019.

[27] 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. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1724–1734, 2014.

[28] Felipe Costa, Sixun Ouyang, Peter Dolog, and Aonghus Lawlor. Automatic generation of natural language explanations. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion*, page 57. ACM, 2018.

[29] Lieven De Lathauwer, Bart De Moor, and Joos Vandewalle. A multilinear singular value decomposition. *SIAM journal on Matrix Analysis and Applications*, 21(4):1253–1278, 2000.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2019.

[31] Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632, 2017.

[32] Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. Unified language model pre-training for natural language understanding and generation. In *Advances in Neural Information Processing Systems*, pages 13063–13075, 2019.

[33] Upol Ehsan and Mark O Riedl. Explainability pitfalls: Beyond dark patterns in explainable ai. *arXiv preprint arXiv:2109.12480*, 2021.

[34] Miao Fan, Chao Feng, Mingming Sun, and Ping Li. Reinforced product metadata selection for helpfulness assessment of customer reviews. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1675–1683, 2019.

[35] Zuohui Fu, Yikun Xian, Ruoyuan Gao, Jieyu Zhao, Qiaoying Huang, Yingqiang Ge, Shuyuan Xu, Shijie Geng, Chirag Shah, Yongfeng Zhang, et al. Fairness-aware explainable recommendation over knowledge graphs. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2020.

[36] Jingyue Gao, Xiting Wang, Yasha Wang, and Xing Xie. Explainable recommendation through attentive multi-view learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3622–3629, 2019.

[37] 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.

[38] Shijie Geng, Zuohui Fu, Yingqiang Ge, Lei Li, Gerard de Melo, and Yongfeng Zhang. Improving personalized explanation generation through visualization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, May 2022.

[39] Azin Ghazimatin, Soumajit Pramanik, Rishiraj Saha Roy, and Gerhard Weikum. Elixir: Learning from user feedback on explanations to improve recommender models. In *Proceedings of the Web Conference 2021*, pages 3850–3860, 2021.

[40] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6645–6649. Ieee, 2013.

[41] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, 2016.

[42] David Gunning, Mark Stefik, Jaesik Choi, Timothy Miller, Simone Stumpf, and Guang-Zhong Yang. Xai - explainable artificial intelligence. *Science Robotics*, 4(37):eaay7120, 2019.

[43] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670. ACM, 2015.

[44] Xiangnan He and Tat-Seng Chua. Neural factorization machines for sparse predictive analytics. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 355–364. ACM, 2017.

[45] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182, 2017.

[46] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250, 2000.

[47] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[48] Min Hou, Le Wu, Enhong Chen, Zhi Li, Vincent W Zheng, and Qi Liu. Explainable fashion recommendation: A semantic attribute region guided approach. In *IJCAI*, 2019.

[49] Jizhou Huang, Wei Zhang, Shiqi Zhao, Shiqiang Ding, and Haifeng Wang. Learning to explain entity relationships by pairwise ranking with convolutional neural networks. In *IJCAI*, pages 4018–4025, 2017.

[50] Robert Jäschke, Leandro Marinho, Andreas Hotho, Lars Schmidt-Thieme, and Gerd Stumme. Tag recommendations in folksonomies. In *European conference on principles of data mining and knowledge discovery*, pages 506–514. Springer, 2007.

[51] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation: n-dimensional tensor factorization for context-aware collaborative filtering. In *Proceedings of the fourth ACM conference on Recommender systems*, pages 79–86. ACM, 2010.

[52] Donghyun Kim, Chanyoung Park, Jinoh Oh, Sungyoung Lee, and Hwanjo Yu. Convolutional matrix factorization for document context-aware recommendation. In *Proceedings of the 10th ACM Conference on Recommender Systems*, pages 233–240. ACM, 2016.

[53] Yong-Deok Kim and Seungjin Choi. Nonnegative tucker decomposition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[54] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1746–1751, 2014.

[55] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[56] Yehuda Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 426–434, 2008.

[57] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[58] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, volume 25, pages 1097–1105, 2012.

[59] Johannes Kunkel, Tim Donkers, Lisa Michael, Catalin-Mihai Barbu, and Jürgen Ziegler. Let me explain: Impact of personal and impersonal explanations on trust in recommender systems. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.

[60] Asher Levi, Osnat Mokryn, Christophe Diot, and Nina Taft. Finding a needle in a haystack of reviews: cold start context-based hotel recommender system.

In *Proceedings of the sixth ACM conference on Recommender systems*, pages 115–122. ACM, 2012.

[61] Lei Li, Li Chen, and Ruihai Dong. Caesar: context-aware explanation based on supervised attention for service recommendations. *Journal of Intelligent Information Systems*, 57:147–170, 2021.

[62] Lei Li, Li Chen, and Yongfeng Zhang. Towards controllable explanation generation for recommender systems via neural template. In *Companion Proceedings of the Web Conference 2020*, pages 198–202, 2020.

[63] Lei Li, Ruihai Dong, and Li Chen. Context-aware co-attention neural network for service recommendations. In *2019 IEEE 35th International Conference on Data Engineering Workshops (ICDEW)*, pages 201–208. IEEE, 2019.

[64] Lei Li, Yongfeng Zhang, and Li Chen. Generate neural template explanations for recommendation. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management*, pages 755–764. ACM, 2020.

[65] Lei Li, Yongfeng Zhang, and Li Chen. Extra: Explanation ranking datasets for explainable recommendation. In *Proceedings of the 44th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 2463–2469, 2021.

[66] Lei Li, Yongfeng Zhang, and Li Chen. On the relationship between explanation and recommendation: Learning to rank explanations for improved performance. *arXiv preprint arXiv:2102.00627*, 2021.

[67] Lei Li, Yongfeng Zhang, and Li Chen. Personalized transformer for explainable recommendation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4947–4957, Online, August 2021. Association for Computational Linguistics.

[68] Lei Li, Yongfeng Zhang, and Li Chen. Personalized prompt learning for explainable recommendation. *arXiv preprint arXiv:2202.07371*, 2022.

[69] Piji Li, Zihao Wang, Lidong Bing, and Wai Lam. Persona-aware tips generation. In *The World Wide Web Conference*, pages 1006–1016, 2019.

[70] Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. Neural rating regression with abstractive tips generation for recommendation. In *Proceedings of the 40th International ACM SIGIR conference on Research and Development in Information Retrieval*, pages 345–354, 2017.

[71] Xueqi Li, Wenjun Jiang, Weiguang Chen, Jie Wu, Guojun Wang, and Kenli Li. Directional and explainable serendipity recommendation. In *Proceedings of The Web Conference 2020*, pages 122–132, 2020.

[72] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*, pages 624–632, 2021.

[73] Yunqi Li, Hanxiong Chen, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. Towards personalized fairness based on causal notion. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1054–1063, 2021.

[74] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. Towards understanding and mitigating social biases in language models. In *International Conference on Machine Learning*, pages 6565–6576. PMLR, 2021.

[75] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[76] Peter J Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. Generating wikipedia by summarizing

long sequences. In *The Sixth International Conference on Learning Representations*, 2018.

[77] Qiang Liu, Shu Wu, Liang Wang, et al. Cot: Contextual operating tensor for context-aware recommender systems. In *Twenty-ninth AAAI Conference on Artificial Intelligence*. AAAI Press, 2015.

[78] Shulin Liu, Yubo Chen, Kang Liu, and Jun Zhao. Exploiting argument information to improve event detection via supervised attention mechanisms. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, 2017.

[79] Tie-Yan Liu. *Learning to rank for information retrieval*. Springer Science & Business Media, 2011.

[80] Yichao Lu, Ruihai Dong, and Barry Smyth. Coevolutionary recommendation model: Mutual learning between ratings and reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 773–782. International World Wide Web Conferences Steering Committee, 2018.

[81] Yichao Lu, Ruihai Dong, and Barry Smyth. Why i like it: Multi-task learning for recommendation and explanation. In *Proceedings of the Eighth ACM Conference on Recommender Systems*. ACM, 2018.

[82] Scott M Lundberg and Su-In Lee. A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, 2017.

[83] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pages 1412–1421, 2015.

[84] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. Explore, exploit, and explain:

personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM conference on recommender systems*, pages 31–39, 2018.

[85] Lei Mei, Pengjie Ren, Zhumin Chen, Liqiang Nie, Jun Ma, and Jian-Yun Nie. An attentive interaction network for context-aware recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*, pages 157–166, 2018.

[86] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013.

[87] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[88] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2007.

[89] Lili Mou, Yiping Song, Rui Yan, Ge Li, Lu Zhang, and Zhi Jin. Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3349–3358, 2016.

[90] Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 188–197, 2019.

[91] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[92] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[93] Georgina Peake and Jun Wang. Explanation mining: Post hoc interpretability of latent factor models for recommendation systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2060–2069, 2018.

[94] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training, 2018.

[95] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[96] Anand Rajaraman and Jeffrey David Ullman. Finding similar items. In *Mining of Massive Datasets*, chapter 3, pages 73–134. Cambridge University Press, 3 edition, 2011.

[97] Steffen Rendle. Factorization machines. In *10th International Conference on Data Mining (ICDM)*, pages 995–1000. IEEE, 2010.

[98] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

[99] Steffen Rendle and Lars Schmidt-Thieme. Pairwise interaction tensor factorization for personalized tag recommendation. In *Proceedings of the third ACM international conference on Web search and data mining*, pages 81–90, 2010.

[100] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. Grouplens: An open architecture for collaborative filtering of netnews.

In *Proceedings of the 1994 ACM conference on Computer supported cooperative work*, pages 175–186, 1994.

[101] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144, 2016.

[102] Herbert Robbins and Sutton Monro. A stochastic approximation method. *The annals of mathematical statistics*, pages 400–407, 1951.

[103] 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.

[104] Masahiro Sato, Budrul Ahsan, Koki Nagatani, Takashi Sonoda, Qian Zhang, and Tomoko Ohkuma. Explaining recommendations using contexts. In *23rd International Conference on Intelligent User Interfaces*, pages 659–664. ACM, 2018.

[105] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 297–305. ACM, 2017.

[106] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. The woman worked as a babysitter: On biases in language generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3407–3412, 2019.

[107] Shaoyun Shi, Hanxiong Chen, Weizhi Ma, Jiaxin Mao, Min Zhang, and Yongfeng Zhang. Neural logic reasoning. In *Proceedings of the 29th ACM*

*International Conference on Information & Knowledge Management*, pages 1365–1374, 2020.

[108] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2219–2228, 2018.

[109] Peijie Sun, Le Wu, Kun Zhang, Yanjie Fu, Richang Hong, and Meng Wang. Dual learning for explainable recommendation: Towards unifying user preference prediction and review generation. In *Proceedings of The Web Conference 2020*, pages 837–847, 2020.

[110] Juntao Tan, Shuyuan Xu, Yingqiang Ge, Yunqi Li, Xu Chen, and Yongfeng Zhang. Counterfactual explainable recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 1784–1793, 2021.

[111] Nava Tintarev and Judith Masthoff. Explaining recommendations: Design and evaluation. In Bracha Shapira, editor, *Recommender Systems Handbook*, chapter 10, pages 353–382. Springer, 2 edition, 2015.

[112] Quoc-Tuan Truong and Hady Lauw. Multimodal review generation for recommender systems. In *The World Wide Web Conference*, pages 1864–1874, 2019.

[113] Ledyard R Tucker. Some mathematical notes on three-mode factor analysis. *Psychometrika*, 31(3):279–311, 1966.

[114] 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.

[115] Nikos Voskarides, Edgar Meij, Manos Tsagkias, Maarten De Rijke, and Wouter Weerkamp. Learning to explain entity relationships in knowledge graphs. In

*Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 564–574, 2015.

[116] Nan Wang, Hongning Wang, Yiling Jia, and Yue Yin. Explainable recommendation via multi-task learning in opinionated text data. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 165–174, 2018.

[117] Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. Tem: Tree-enhanced embedding model for explainable recommendation. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1543–1552. International World Wide Web Conferences Steering Committee, 2018.

[118] Xiang Wang, Dingxian Wang, Canran Xu, Xiangnan He, Yixin Cao, and Tat-Seng Chua. Explainable reasoning over knowledge graphs for recommendation. In *Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019.

[119] Xiting Wang, Yiru Chen, Jie Yang, Le Wu, Zhengtao Wu, and Xing Xie. A reinforcement learning framework for explainable recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 587–596. IEEE, 2018.

[120] Zhongqing Wang and Yue Zhang. Opinion recommendation using a neural model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1626–1637, 2017.

[121] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, 2018.

[122] Yikun Xian, Zuohui Fu, S Muthukrishnan, Gerard De Melo, and Yongfeng Zhang. Reinforcement knowledge graph reasoning for explainable recommen-

dation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 285–294, 2019.

[123] Yikun Xian, Zuohui Fu, Handong Zhao, Yingqiang Ge, Xu Chen, Qiaoying Huang, Shijie Geng, Zhou Qin, Gerard De Melo, Shan Muthukrishnan, et al. Cafe: Coarse-to-fine neural symbolic reasoning for explainable recommendation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1645–1654, 2020.

[124] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. Attentional factorization machines: Learning the weight of feature interactions via attention networks. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 2017.

[125] Xin Xin, Bo Chen, Xiangnan He, Dong Wang, Yue Ding, and Joemon Jose. Cfm: convolutional factorization machines for context-aware recommendation. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3926–3932. AAAI Press, 2019.

[126] Xuhai Xu, Ahmed Hassan Awadallah, Susan T. Dumais, Farheen Omar, Bogdan Popp, Robert Rounthwaite, and Farnaz Jahanbakhsh. Understanding user behavior for document recommendation. In *Proceedings of The Web Conference 2020*, pages 3012–3018, 2020.

[127] Yang Yang, Junmei Hao, Canjia Li, Zili Wang, Jingang Wang, Fuzheng Zhang, Rao Fu, Peixu Hou, Gong Zhang, and Zhongyuan Wang. Query-aware tip generation for vertical search. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2893–2900, 2020.

[128] Lili Yao, Yaoyuan Zhang, Yansong Feng, Dongyan Zhao, and Rui Yan. Towards implicit content-introducing for generative short-text conversation systems. In *Proceedings of the 2017 conference on empirical methods in natural language processing*, pages 2190–2199, 2017.

[129] 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.

[130] Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, 2020.

[131] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *Foundations and Trends® in Information Retrieval*, 14(1):1–101, 2020.

[132] 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.

[133] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 83–92, 2014.

[134] Yongfeng Zhang, Haochen Zhang, Min Zhang, Yiqun Liu, and Shaoping Ma. Do users rate or review? boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1027–1030, 2014.

[135] Lujun Zhao, Kaisong Song, Changlong Sun, Qi Zhang, Xuanjing Huang, and Xiaozhong Liu. Review response generation in e-commerce platforms with

external product information. In *The world wide web conference*, pages 2425–2435, 2019.

[136] Lei Zheng, Vahid Noroozi, and Philip S Yu. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 425–434, 2017.

[137] Yinhe Zheng, Rongsheng Zhang, Minlie Huang, and Xiaoxi Mao. A pre-training based personalized dialogue generation model with persona-sparse data. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9693–9700, 2020.

[138] 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.

[139] Yaxin Zhu, Yikun Xian, Zuohui Fu, Gerard de Melo, and Yongfeng Zhang. Faithfully explainable recommendation via neural logic reasoning. In *2021 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 2021.

[140] Lixin Zou, Shengqiang Zhang, Hengyi Cai, Dehong Ma, Suqi Cheng, Shuaiqiang Wang, Daiting Shi, Zhicong Cheng, and Dawei Yin. Pre-trained language model based ranking in baidu search. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 4014–4022, 2021.

# List of Publications

1. Lei Li, Yongfeng Zhang, Li Chen, *Personalized Prompt Learning for Explainable Recommendation*, **ACM Transactions on Information Systems**, 2022. [*submitted*]

2. Lei Li, Yongfeng Zhang, Li Chen, *On the Relationship between Explanation and Recommendation: Learning to Rank Explanations for Improved Performance*, **ACM Transactions on Intelligent Systems and Technology**, 2022. [*submitted*]

3. Shijie Geng, Zuohui Fu, Yingqiang Ge, Lei Li, Gerard de Melo, Yongfeng Zhang, *Improving Personalized Explanation Generation through Visualization*, **Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics**, Dublin, Ireland, May 22–27, 2022.

4. Zhirun Zhang, Li Chen, Tonglin Jiang, Yutong Li, Lei Li, *Effects of Feature-based Explanation and Its Output Modality on User Satisfaction with Service Recommender Systems*, **Frontiers in Big Data**, volume 5, article 897381, pages 1-17, May 2022.

5. Lei Li, Yongfeng Zhang, Li Chen, *Personalized Transformer for Explainable Recommendation*, **Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing**, pages 4947-4957, Online, Thailand, August 1–6, 2021. (**oral paper**)

6. Lei Li, Li Chen, Ruihai Dong, *CAESAR: Context-Aware Explanation based*

*on Supervised Attention for Service Recommendations*, **Journal of Intelligent Information Systems**, volume 57 (1), pages 147-170, August 2021.

7. Lei Li, Yongfeng Zhang, Li Chen, *EXTRA: Explanation Ranking Datasets for Explainable Recommendation*, **Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval**, pages 2463-2469, Virtual Event, Canada, July 11–15, 2021.

8. Lei Li, Yongfeng Zhang, Li Chen, *Generate Neural Template Explanations for Recommendation*, **Proceedings of the 29th ACM International Conference on Information & Knowledge Management**, pages 755-764, Virtual Event, Ireland, October 19–23, 2020.

9. Lei Li, Li Chen, Yongfeng Zhang, *Towards Controllable Explanation Generation for Recommender Systems via Neural Template*, **Companion Proceedings of the Web Conference 2020**, pages 198-202, Taipei, Taiwan, April 20–24, 2020.

10. Lei Li, Ruihai Dong, Li Chen, *Context-aware Co-Attention Neural Network for Service Recommendations*, **2019 IEEE 35th International Conference on Data Engineering Workshops**, pages 201-208, Macao, China, April 8–12, 2019.

# CURRICULUM VITAE

Academic qualifications of the thesis author, Mr. LI Lei:

- Received the degree of Bachelor of Engineering from Shenzhen University, June 2017.
- Received the degree of Bachelor of Science from Shenzhen University, June 2017.

May 2022