Towards Transparent and Fair Personalization Systems

A

Dissertation

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

in partial fulfillment

of the requirements for the degree

Doctor of Philosophy

by

Nan Wang

May 2023

APPROVAL SHEET

This

Dissertation

is submitted in partial fulfillment of the requirements
for the degree of

Doctor of Philosophy

Author: Nan Wang

This Dissertation has been read and approved by the examing committee:

Advisor:	Hongning Wang
Advisor:	
Committee Member:	Aidong Zhang
Committee Member:	Yangfeng Ji
Committee Member:	Sheng Li
Committee Member:	Yongfeng Zhang
Committee Member:	
Committee Member:	

Accepted for the School of Engineering and Applied Science:

[illegible]

Jennifer L. West, School of Engineering and Applied Science

May 2023

Abstract

Personalization systems (PS) are applied in nearly every corner of the internet through recommendation, content supply, messaging, and so on. Not only do business owners depend on personalization for recommending the relevant items to the right users, but also consumers need personalization to find useful information without being overwhelmed in the info-times. Due to these reasons, enormous efforts have long been devoted in developing more powerful AI and machine learning techniques to improve the performance of PS.

In recent years, however, people start to realize that PS empowered by these techniques may lead to undesired effects and undermine the original good purposes of personalization. One of the main issues with PS is their lack of transparency, which means that users may not fully understand why they are being recommended certain items or content, which can lead to confusion and mistrust. Another issue is fairness. Due to the vast amounts of data used to train these systems, PS can inadvertently learn and perpetuate biases that exist in society. This can have serious consequences, such as perpetuating systemic inequality and discrimination. Addressing the issues of transparency and fairness in PS is essential for ensuring that they are trusted and effective tools to continue to benefit users, businesses, and society as a whole.

This dissertation focuses on improving the transparency and fairness of PS, which contributes to the development of more responsible PS that benefit all stakeholders. To enhance transparency, I propose to generate intuitive, textual explanations for personalized results. The explanations are expected to help users make more informed decisions and build trust in the system. For fairness, I investigate and propose algorithms to address the issues from different perspectives in PS. In general, the algorithms aim to generate personalized results without discrimination in serving users and business owners with different social constructs. Comprehensive and rigorous analysis and experiments demonstrates the approaches' effectiveness in various contexts and applications.

Acknowledgement

During my Ph.D. studies, I feel incredibly fortunate to have received tremendous support from a diverse range of people, both within academia and industry. I would like to take this opportunity to express my sincere gratitude to everyone who has contributed to this unique journey for my life. Without their encouragement and assistance, completing this dissertation would have been impossible.

First and foremost, I cannot overstate my gratitude to my advisor, Prof. Hongning Wang, for his exceptional guidance and mentorship throughout my Ph.D. studies. His instruction taught me to be a rigorous and insightful researcher, constantly challenging me to think critically and develop the skills necessary for success in my field. There are numerous challenges and obstacles during my Ph.D. study, and Prof. Wang was always available to provide guidance and support, helping me navigate through the toughest of times, for which I am profoundly grateful. I recognize how fortunate I am to have had him as my advisor and mentor, and I am confident that the knowledge and skills I have acquired under his tutelage will benefit me greatly throughout my future career.

I would like to thank my dissertation committee members Prof. Aidong Zhang, Prof. Yangfeng Ji, Prof. Sheng Li, and Prof. Yongfeng Zhang for their invaluable suggestions on my dissertation. Prof. Aidong Zhang's comments helped me to think more rigorously in evaluating the impacts of my research in real-world applications. Discussing with Prof. Yangfeng Ji and Sheng Li is always inspiring. Their critical thinking and insightful views guide me to further improve my approaches. Prof. Yongfeng Zhang is a pioneer in explainable personalization. His works and suggestions are consistently inspiring for my research and helping me to broaden the research scope. I vividly recall his 2014 paper on explainable recommendation as the first piece I read in this field that ultimately led me to pursue this research area.

I would also like to extend my appreciation to my great collaborators during my Ph.D study. Their contributions have been instrumental in helping me develop my research ideas, refine my methodology, and produce high-quality research. Their feedback and insights have been invaluable to me, and I am grateful for their support and collaboration. I thank Yiling Jia, Aobo Yang, Lu Lin, Jundong Li and many others at University of Virginia for great collaboration. I had several fantastic internship experiences in the industry. I would like to thank Zhen Qin, Le Yan, Xuanhui Wang, Michael Bendersky, Maryam Karimzadehgan, Branislav Kveton and others at Google for their invaluable mentorship and guidance in proposing and conducting research to address real-world challenges. I also want to thank Shaoliang Nie, Qifan Wang, Hamed Firooz and other collaborators at Meta for providing me with a memorable internship before graduation. Many thanks all members of our HCDM group for building a great atmosphere in research. I really enjoyed having discussions and group activities in the group.

Last but not least, I would like to express my thanks to my family for their unconditional love and support. Although they are on the other side of the earth in China, my family has been a constant source of encouragement, providing me with the strength and motivation to persevere through challenging times and overcome the obstacles that came my way. Their unwavering belief in me and my abilities gave me the confidence to pursue my dreams and achieve my goals.

Contents

List of Tables

List of Figures

Chapter 1

Introduction and Overview

Personalization systems (PS) are ubiquitous in today's online ecosystem, providing users with tailored recommendations, content, and messaging [1–8]. They have become essential tools for e-commerce merchants seeking to target their products to the right users, as well as for consumers seeking to navigate the vast amounts of information available on the internet. However, the increasing use and power of PS has led to concerns about the unintended consequences of their deployment. As PS techniques become more sophisticated, it is becoming apparent that they can have negative impacts on users, items, producers, platforms, and society at large, potentially undermining the original intent of personalization [9].

First of all, non-transparency and ambiguity in recommendations can reduce their effectiveness and erode users' trust in the system. Furthermore, when personalization services are offered globally or in culturally diverse areas, there is a risk of unfair treatment based on gender, ethnicity, or other factors, which can discourage users from using the system and raise legal concerns. These issues have also caught the attention of policy makers. The General Data Protection Regulation (GDPR) implemented in 2018 [10], addresses that users have a "right to an explanation" for an output of the algorithm. In 2016, the Obama administration urged researchers to analyze "how technologies can deliberately or inadvertently perpetuate, exacerbate, or mask discrimination." Therefore, transparency and fairness are two critical properties for PS to expand their applications in different scenarios and consistently benefit everyone.

It is essential to develop personalization techniques that take into account the above mentioned issues and work towards more transparent and fair PS. However, due to complexity of these algorithms and the obscurity of user preferences learned from data, addressing these issues can be challenging. Firstly, many personalization systems are build on complex models, such as neural networks, which do not allow easy interpretation of the reasons of the output personalization results [11, 12]. Besides, there are different ways to improve transparency for end users. For instance, one can teach the system users the technique used, which may increase their understanding of how the system works. Another approach is to provide intuitive explanations along with the personalization results that help them understand the results [13–15]. The effectiveness of these different methods may vary depending on the context and user needs. Secondly, PS are usually trained on large amount of data, from which the model learns the implicit patterns and correlations to generate personalized results. The fairness issues may occur at different stages of the model development [16–22], such as during data processing, training, or serving. More importantly, fairness is a subjective concept and has different definitions [23–25], largely depending on the application and the system designers' requirements. Thus, we need to develop techniques that cover and generalize to various fairness needs.

In this dissertation, I aim to improve the transparency and fairness of PS, while maintaining high personalization performance. For transparency, after extensive amount of research effort endeavored to advance the personalization algorithms [26–29], solutions that explain the machine-made decisions have recently come under the spotlight [14, 30]. Numerous studies have shown that explanations help users make more accurate decisions, improve their acceptance of recommendations [14], and build up trust towards the personalization system [15]. To make sure that the explanations are helpful and users can easily understand and enjoy interact with them, I propose to generate personalized, intuitive textual explanations that follow the following four principles: *informative*, *faithful*, *readable*, and *comparable*. I developed specific methods and algorithms to target each of the four principles, accordingly. For fairness, there exist various fairness definitions for different problems and different stakeholders in the two-sided markets [31]. They can be roughly classified into two categories: (1) *universal fairness*, meaning the personalized results should be totally independent from the sensitive attribute of the target users/items, (2) *measure-specific fairness*, that defines the fairness requirements according to a specific metric. To promote fairness in personalized results, I firstly propose a framework for learning unbiased user embeddings, leading to universal fairness in downstream tasks that use the resulting embeddings for personalization. Secondly, I also propose to realize metric-specific fairness in the personalized explanation generation, ensuring that certain aspects of the generated explanations does not discriminate against the users'/items' protected attributes.

1.1 Challenges and Insights

1.1.1 Generating Intuitive Explanations for Personalized Results

Tremendous amount of effort has been devoted onto improving the effectiveness of personalization algorithms. Promising results have been reported and many algorithms have been successfully deployed in practical systems. However, one fundamental question that has not received enough attention in previous studies is *how a system should explain those customized results to its users* [32]. Modern personalized information systems are black boxes to their users: computerized oracles give advice, but cannot be questioned. The lack of transparency in PS [15] leaves users in a dilemma: a user can only assess the quality of personalized results by taking the suggested items, e.g., read the recommended articles; however, in order for him/her to adopt the system's customized results, he/she needs to first build trust over the system. As a result, the lack of explanation for the personalized results has prevented acceptance of personalization techniques in many important but high-risk domains, e.g., healthcare, education and finance industries.

Previous research has shown that explanations help people make more accurate decisions [13], improve user acceptance of recommendations [14], and increase trust in the information systems [15]. However, explaining personalized outputs to users is non-trivial. Presumably, the most straightforward explanation is the direct illustration of a system's employed personalization algorithm (e.g., how the latent factors are calculated in a neural network model). However, rich background knowledge about the algorithm is required for users to understand such explanations. This type of explanations are too abstruse for an average user to understand, nor to help them in decision making. Instead, we aim to provide personalized textual explanations for users. This goal imposes a even higher requirement on personalization, e.g., when searching for diabetes medications, different users might hold distinct decision criteria: e.g., price vs. side-effect. Therefore, they will need different explanations in assessing the utility of personalized results. To generate useful explanations that support their decision-making, the first challenge is to to precisely understand users' preferences. On top of personalizing the explanation generation, we also require the explanations to be informative, faithful, readable, and comparable.

Informative: Explainable recommendation via multi-task learning in opinionated text data. Arguably, the most important value of explanations in an information system is not to convince users to adopt customized results (i.e., promotion), but to allow them to make more informed and accurate decisions about which results to utilize (i.e., satisfaction) [13]. Therefore, the explanations need to be informative to assist users' decision-making process. One approach to achieve this is to predict the opinionated content that users would provide on the recommended item. This could include how they would comment on specific features of the item, which can serve as an informative explanation to illustrate why users should pay attention to those recommended items. To this end, we develop a joint tensor factorization solution to integrate two complementary tasks of user preference modeling for recommendation and opinionated content modeling for explanation, i.e., a multi-task learning approach. With the available user opinionated reviews, rich information can be extracted with feature-level sentiment analysis techniques [33, 34]. The explanations can be generated by filling the predicated features and corresponding opinionated descriptions in predefined templates.

Faithful: Explainable recommendation with factorization tree. If users are persuaded to accept personalized results that are subsequently found to be inferior, their confidence and trust in the system will rapidly deteriorate [14, 35]. Hence, the fidelity of explanations becomes a prerequisite for explainable PS. The fidelity of explanation and the quality of personalization have long been considered as irreconcilable [11]: one has to trade personalization quality for explainability. Various solutions have been proposed to *approximate* the underlying personalization mechanisms for explanation [11, 12]. But to what extent these approximated explanations comply with the personalization models is unknown, i.e., no guarantee in explanation fidelity. Prior studies show that rule-based explanations are more easy to perceive and justify by the end-users [36]. Motivated by this, we propose to integrate the rule-based decision making into the learning of latent factors. More specifically, we treat the latent factors as a function of the rules: based on different outcomes of the rules, the associated groups of users and items should be routed to the designated latent factors, which are then optimized for recommendation. As the latent factors are learned subject to the explanation rules, the fidelity of explanations is ensured; and because latent factors are optimized for improving personalization, the quality of personalization is also provided.

Readable: sentiment aligned natural language explanation generation. PS serve at the frontier of Human-centered AI research and works as the bridge between humans and AI. Therefore, the explanations in PS need to be readable such that users can easily understand them and enjoy interacting with the system. Free-form natural language based explanations have been identified as a preferred medium for explaining the recommendations [30, 37–39]. But one need to note that the explanations generated with fairly fluent language are harder to control, i.e., the alignment between explanations and personalized results is not guaranteed. We believe the sentiment delivered by the explanation text needs to reveal the details of how items are scored and ranked differently by the system. We formulate this as sentiment alignment between the explanation text and system's corresponding recommendation [38]. In particular, the learning of recommendation is modeled as a neural collaborative filtering problem, and the learning of explanation is modeled as a neural text generation problem. We force the recommendation module to directly influence the learning of explanations to enforce sentiment alignment in both training and inference time for improved explainable recommendation.

Comparable: generating comparative explanations of recommendations. Users make decisions based on comparisons between different options, and thus explanations should ideally assist users in making such comparisons. However, existing explainable recommendation solutions are not optimized to help users make such comparative decisions for two major reasons. In this part of work, we focus on explaining how one item is compared with another; then by using a commonly shared set of items as references (e.g., items the user has reviewed before), the comparisons among the recommended items emerge [40]. Our solution is designed to generically work on top of other existing recommender systems. We do not have any assumptions about how the recommendation algorithm ranks items, but only require it to provide a ranking score for each item to our model (i.e., ordinal ranking) which reflects a user's preference over the recommended item. This makes our solution readily applicable to explain plenty of effective recommendation algorithms deployed in practice.

1.1.2 Promoting Fairness in Personalization Systems

There are increasing concerns on the issues of biased or unfair personalization results. Existing studies show that PS can be vulnerable to unfairness in several aspects, which may lead to detrimental effects for users in certain demographic groups. For example, in e-commerce systems, PS may promote items that mainly maximize the profit of certain producers. In online job marketplaces, RS may lead to racial or gender discrimination by disproportionately recommending low-payment jobs to certain user groups. When the users realize the bias or unfairness, it can also adversarially affect the users' reliance and trust in the system [41]. From a broader perspective, the bias and unfairness presented in personalization results can reinforce the stereotype by influencing how users behave online, and jeopardizes fairness in society in the long run [42–44].

In recent years, there have been considerable efforts from both academia and industry to mitigate fairness issues in PS [45–50]. However, the concept of fairness is subjective and entirely depends on the application and the fairness requirements established by the system designer. As a result, various fairness types have been defined from different perspectives in the literature [23–25, 51–56], such as group, individual, casual, and associative fairness, which usually need specifically designed algorithms to achieve them. Moreover, since PS operate in two-sided markets [31] where both users and business owners should be subject to fairness considerations, it is essential to address fairness from both perspectives. In this dissertation, we investigate two categories of fairness: universal fairness and measure-specific fairness, and propose algorithms for each. Specifically, for universal fairness, we require that personalized results should not depend on the sensitive attributes of the users/items subject to the results. In other words, we aim to enforce independence between the outputs and the sensitive attributes under consideration. In measure-specific fairness, we instead aim to improve fairness w.r.t. a specific fairness metric, defined on specific aspects of the results.

Universal fairness by learning unbiased user and item embeddings. User and item embedding learning in social networks is an indispensable building block in many personalization systems [57–61]. Network embedding methods map each user to a low-dimensional embedding vector that reflects the users' structural information from the observed connections, which are then employed to solve downstream tasks, such as friend recommendation in social networks or user interest prediction in e-commerce platforms [62, 63]. Since the formation of a graph is inevitably affected by certain *sensitive node attributes*, the node embeddings can inherit such sensitive information and introduce undesirable biases in downstream tasks. To tackle this problem, we propose a principled new way for unbiased user embedding by learning node embeddings from an *underlying bias-free graph*, which is not influenced by sensitive user attributes, such that downstream personalization tasks using these embeddings will be fair by generating results independent from users'/items' sensitive attributes.

Optimizing measure-specific fairness in explanation generation. Natural language explanation generation [40, 64, 65] is essentially a personalized text generation problem [66–69]. The generators are usually trained on user written text, e.g., reviews collected on e-commerce platforms. However, due to historical, social, or behavioral reasons, bias may exist that associates certain properties of user-written text, such as sentiment or detailedness, with users' or items' protected attributes like users' gender, age, or items' production country. This bias can be inherited by generators, resulting in discriminative text generation concerning users' or items' protected attributes [43, 44, 48]. In this dissertation, we investigate the fairness in personalized text generation in explainable recommendation, and propose two perspectives in achieving fairness: counterfactual fairness on individuals [70], and strong demographic parity for group-wise fairness. We then propose frameworks for achieving measure-specific fairness from these two perspectives.

1.2 Dissertation Structure

The rest of this dissertation is structured as follows: In Chapter 2, we introduce solutions to enhance the transparency of personalization systems by providing intuitive textual explanations. We follow the four principles in proposing explanation generation methods: informative, faithful, readable, and comparable. In Chapter 3, we investigate fairness in personalized results from different perspectives and develop solutions accordingly. We propose an unbiased graph embedding method for learning unbiased user and item representations from online networks, which are then utilized in downstream tasks for universal fairness. Additionally, we propose methods for counterfactual fairness and group-wise fairness in personalized explanation generation, respectively. In Chapter 4, we summarize this dissertation and discuss future research directions.

Chapter 2

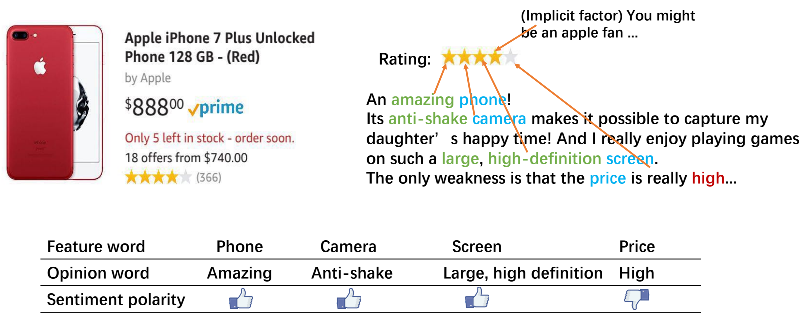
Improving Transparency by Providing Intuitive Textual Explanations

Modern personalization systems are mostly black boxes to their users: computerized oracles give advice, but cannot be questioned. The lack of transparency in personalization [15] leaves users in a dilemma: a user can only assess the quality of personalized results by taking the suggested instances, e.g., read the recommended articles; however, in order for him/her to adopt the system's customized results, he/she needs to first build trust over the system. Explaining the automatically generated recommendations would bridge the gap. Arguably, the most important contribution of explanations is not to convince users to accept the customized results (i.e., promotion), but to allow them to make more informed and accurate decisions about which results to utilize (i.e., satisfaction) [71].

Existing recommendation algorithms emphasize end-to-end optimization of performance metrics, such as Root-Mean-Square Error and Normalized Discounted Cumulative Gain, which are defined on numerical ratings or ranking orders reflecting a user's overall preference over a set of items. Various algorithms [27, 72–77] have been proposed to optimize those metrics. However, it is known that humans are complex autonomous systems: a click/purchase decision is usually a composition of multiple factors. The end-to-end learning scheme can hardly realize the underlying reasons behind a user's decision making process. As a result, although such algorithms achieve great success in quantitative evaluations, they are still computerized oracles that merely give advice, but cannot be questioned, especially when a new or wrong recommendation has been made. This greatly limits the practical value of personalization systems.

In this chapter, we introduce solutions for explaining personalized recommendation results by focusing on different perspectives of explanations, i.e., informativeness, fidelity, readability, and comparability. In Section 2.1, we propose a multi-task learning framework called MTER for user preference modeling and opinionated content model. MTER can generate informative opinionated explanations that explain why each item feature may match the user's interest. In Section 2.2, we propose a rule-based explanation generation method, where we integrate rule-based decision making into the learning of latent factor models for faithful explanation generation. Both MTER and FacT generate template-based explanations that rely on mining feature and opinion words to fill in pre-defined explanation templates. While effective, these explanations lack the expressiveness and diversity of natural language that users employ daily. Moreover, the templates may not be suitable for all cases of explanation needs, and different applications require a unique set of meticulously designed templates. Such robotic style explanations are usually considered less appreciated by users. To enhance the expressiveness and flexibility of explanation generation, we move beyond predicting keywords in templates and instead focus on natural language explanations. Specifically, we propose a two-pronged approach: in Section 2.3, we prioritize improving the faithfulness of explanations by aligning the sentiment of the generated explanations with the predicted rating scores of each item. Then in Section 2.4, we introduce a comparative explanation generation framework that explains the comparisons between two items being presented as recommendations.

2.1 Explainable Recommendation via Multi-Task Learning in Opinionated Text data


Apple iPhone 7 Plus Unlocked Phone 128 GB - (Red)
by Apple
$888^{00} prime
Only 5 left in stock - order soon.
18 offers from $740.00
(366)
(Implicit factor) You might be an apple fan ...
Rating:

An amazing phone!
Its anti-shake camera makes it possible to capture my daughter' s happy time! And I really enjoy playing games on such a large, high-definition screen.
The only weakness is that the price is really high...

Feature word	Phone	Camera	Screen	Price
Opinion word	Amazing	Anti-shake	Large, high definition	High
Sentiment polarity	👍	👍	👍	👎

Figure 2.1: Composition of star ratings in a typical user review.

We argue that a good recommendation algorithm should consist of companion learning tasks focusing on different aspects of users' decisions over the recommended items, such that the observed final decisions (e.g., clicks or ratings) can be mutually explained by the associated observations. In this work, we focus on opinionated review text that users provide in addition to their overall assessments of the recommended items, and aim to exploit such information to enhance and explain the recommendations. Figure 2.1 illustrates how opinionated content in a user's review reflects the composition of his/her overall assessment (rating) of the item. For this given four-star overall rating, three positively commented features contribute to his/her positive assessment, and one negatively commented feature explains his/her negative judgment. If an algorithm could learn to predict not only the user's overall rating, but also the opinionated comment he/she would provide on this item (e.g., how would he/she endorse the features of this item), the recommendation will become self-explanatory. And clearly these two predictions are not independent: the predicted overall assessment has to be supported by the predicted opinionated comments. Therefore, the additional information introduced by the companion content modeling task would help improve the quality of recommendation task.

Considerable effort has been devoted to utilizing user-generated opinionated content for providing text-based explanations. One type of solutions leverage phrase-level sentiment analysis [78, 79], which zoom into users' detailed feature-level opinions to explain the recommendations. But these solutions simply map feature-level comments into numeric ratings, and thus ignore the detailed reason that the user likes/dislikes the feature of a particular item. It is impossible for such type of algorithms to explain how exactly the highlighted features of their recommendations match the user's specific preference, which inevitably lose detailed opinion information on each feature.

In this work, We focus on explaining factorization-based recommendation algorithms [72, 80] by taking a holistic view of item recommendation and sentiment analysis. We develop a joint tensor factorization solution to integrate two complementary tasks of *user preference modeling for recommendation* and *opinionated content modeling for explanation*, i.e., a multi-task learning approach [81–83]. Extensive experimental comparisons between our proposed solution and existing explainable recommendation algorithms demonstrate the effectiveness of our solution in both item recommendation and explanation generation tasks in two different application scenarios.

2.1.1 Related Work

Latent factor models, such as matrix factorization [72] and tensor factorization [80], have achieved great success in practical personalization systems. This type of algorithms map users and recommendation candidates to a lower dimensional latent space, which encodes affinities between different entities. Despite their promising recommendation quality, the latent and nonlinear characteristics of this family of solutions make it frustratingly difficult to explain the generated recommendations.

The lack of interpretability in factorization-based algorithms has attracted increasing attention in the research community. Zhang et al. [78] combined techniques for phrase-level sentiment analysis with matrix factorization.

Abdollahi and Nasraoui [11] introduced explainability as a constraint in factorization: the learnt latent factors for a user should be close to those learnt for the items positively rated by him/her. However, such type of algorithms only explain ratings, either the overall rating or feature-level ratings, while ignore the details in a user's comment. They are restricted to some generic explanations, such as "You might be interested in [feature], on which this product performs well" [78]. Our work introduces a companion learning task of opinionated content modeling, in parallel with the task of factorization based recommendation. We explicitly model how a user describes an item's features with latent factors, so that we can explain why he/she should pay attention to a particular feature of a recommended item, e.g., "We recommend this **phone** to you because of its *high-resolution* **screen**."

There are also solutions considering the latent factor models from a probabilistic view, which provides the flexibility of modeling associated opinionated text data for explanation. Wang and Blei [84] combine probabilistic matrix factorization with topic modeling for article recommendation. Explanations are provided by matching topics in items against the target user. A follow-up work [85] introduces aspect-level topic modeling to capture users' finer-grained sentiment on different aspects of an item, so that aspect-level explanations become possible. Ren et al. [79] introduce social relations into topic modeling based recommendation via a concept named viewpoint, which enables social explanation. However, the probabilistic modeling of latent factors is usually limited by the feasibility of posterior inference, which restricts the choices of distributions for modeling the rating and text content. And the resolution of explanations is often confined at the topic level, which leads to generic explanation across all users. Our solution directly works with factorization-based latent factor models to capture a more flexible dependency among user, item and the associated opinionated content. Via a joint tensor factorization, latent representation of each opinionated phrase in the vocabulary is learnt for generating personalized context-aware explanations.

2.1.2 Joint Tensor Factorization for Explainable Recommendation

In this section, we elaborate our multi-task learning solution for explainable recommendation. We exploit the opinionated review text data that users provide in addition to their overall assessments of the recommended items to enhance and explain the recommendation. Two companion learning tasks, i.e., *user preference modeling for recommendation* and *opinionated content modeling for explanation*, are integrated via a joint tensor factorization.

In the following discussions, we denote m, n, p, q as the number of users, items, features and opinionated phrases in a dataset, and a, b, c, d as the corresponding dimensions of latent factors for them in the learnt model. As a result, after factorization, users, items, features and opinion phrases can be represented by four non-negative matrices $U \in \mathbb{R}_+^{m\times a}$, $I \in \mathbb{R}_+^{n\times b}$, $F \in \mathbb{R}_+^{p\times c}$ and $O \in \mathbb{R}_+^{q\times d}$ in the latent factor space, respectively. Note that these four types of entities are associated with different degrees of complexity in practice, and therefore we do not restrict them to the same dimension of latent factors. To capture users' detailed feature-level opinions, we assume the existence of a domain-specific sentiment lexicon $\mathcal{L}$. Each entry of $\mathcal{L}$ takes the form of (feature, opinion, sentiment polarity), abbreviated as (f, o, s), to represent the sentiment polarity s inferred from an opinionated text phrase o describing feature f. Specifically, we label the sentiment polarity s as positive (+1) or negative (-1), but the developed solution can be seamlessly extended to multi-grade or continues rating cases. Based on this notation, our sentiment analysis is to map each user's review into a set of (f, o, s) entries. We use R_i^U and R_j^I to denote the set of reviews associated with user i and item j, respectively.

User Preference Modeling for Item Recommendation

This task is to predict the relevance of a recommendation candidate to a user, such that relevant candidates can be ranked higher. Traditional solutions perform the estimation by mapping users and items to a shared latent space via low-rank factorization over an input user-item affinity matrix [76, 86]. However, because this input matrix is usually constructed by users' overall assessment of items, such as clicks or ratings, the learnt factors cannot differentiate nor explain the detailed reason that a user likes/dislikes an item. To address this limitation, we focus on feature-level preference modeling for item recommendation. We aim to not only predict a user's overall assessment of an item, but also his/her preference on each feature of this item to enhance the recommendation.

Since different users would focus on different features of the same item, and even for the same feature of an item, different users might express distinct opinions on them, we use a three-way tensor $X \in \mathbb{R}_+^{m \times n \times p}$ to summarize such a high-dimensional relation. The key is to define the element X_{ijk} in this tensor, which measures to what extent user i appreciates item j's feature k reflected in his/her opinionated review set R_i^U. In this work, we adopt the method developed in [33] to construct a domain-specific sentiment lexicon $\mathcal{L}$ for analyzing users' detailed feature-level opinions. As the construction of a sentiment lexicon is not a contribution of this work and limited by space, we will not discuss the details of this procedure; interested readers can refer to [33, 78] for more details.

Based on the constructed sentiment lexicon $\mathcal{L}$, a user review can be represented as a list of (f, o, s) tuples. It is possible that a user mentions a particular feature multiple times in the same review but with phrases of different sentiment polarities. To denote the overall sentiment, we calculate the summation of all sentiment polarities that user i has expressed on item j's feature k. Suppose feature k is mentioned t_{ijk} times by user i about item j with the sentiment polarity labels $\{s_{ijk}^1, s_{ijk}^2, \ldots, s_{ijk}^{t_{ijk}}\}$, we define the resulting feature score as $\widehat{s}_{ijk} = \sum_{n=1}^{t_{ijk}} s_{ijk}^n$.

As we discussed in the introduction, a user's overall assessment of an item is usually a composition of multiple factors. In order to build the connection between a user's feature-level and overall assessments of an item, we introduce the overall assessment as a dummy feature to all items and append the overall rating matrix $A \in \mathbb{R}_+^{m \times n}$ to tensor X. This results in a new tensor $\widetilde{X} \in \mathbb{R}_+^{m \times n \times (p+1)}$. To normalize the scale between feature score $\widehat{s}_{ijk}$ and item overall rating A_{ij} in $\widetilde{X}$, we perform the following nonlinear mapping on its elements introduced by the feature scores,

$$\widetilde{X}_{ijk} = \begin{cases} 0, \text{if } f_k \text{ is not mentioned by } u_i \text{ about } i_j \\ 1 + \frac{N-1}{1+\exp(-\widehat{s}_{ijk})}, \quad \text{otherwise} \end{cases} \tag{2.1.1}$$

where N is the highest overall rating in the target domain.

Tensor $\widetilde{X}$ describes the observed affinity among users, items and features in a training data set. To predict unknown affinity among these three types of entities in testing time, we factorize $\widetilde{X}$ in a lower dimensional space to find the latent representation of these entities, and complete the missing elements in $\widetilde{X}$ based on the learnt representations. As we do not restrict these three types of entities to the same dimension of latent factors, we require a more flexible factorization scheme. Tucker decomposition [80, 87] best fits for this purpose, i.e.,

$$\begin{aligned} &\min_{\widehat{X}} \ ||\widehat{X} - \widetilde{X}||_F \qquad (2.1.2) \\ &\text{s.t. } \widehat{X} = \sum_{r=1}^{a}\sum_{t=1}^{b}\sum_{v=1}^{c} g_{rtv}\mathbf{u}_r \otimes \mathbf{i}_t \otimes \mathbf{f}_v \\ &\forall r, t, v \quad \mathbf{u}_r \geq 0, \mathbf{i}_t \geq 0, \mathbf{f}_v \geq 0, \text{and } g_{rtv} \geq 0 \end{aligned}$$

where $\mathbf{u}_r$ is the r-th column in the resulting user factor matrix U, $\mathbf{i}_t$ is the t-th column in the resulting item factor matrix I, $\mathbf{f}_v$ is the v-th column in the resulting feature factor matrix $\widetilde{F}$ (with dummy overall assessment feature expansion), $|| \cdot ||_F$ denotes the Frobenius norm over a tensor, and $\otimes$ denotes vector outer product. As we have mapped the feature scores to the same range of overall ratings in the target domain (i.e., $[1, N]$), we impose non-negative constraint over the learnt latent factors to avoid any negative predictions.

In Tucker decomposition, a core tensor $\mathcal{G} \in \mathbb{R}_+^{a \times b \times c}$ is introduced to describe how and to what extent different tensor elements interact with each other. This provides us another degree of freedom in performing the joint factorization in our multi-task learning solution. We will carefully elaborate this important advantage later when we discuss the detailed learning procedure in Section 2.1.2.

The resulting factor matrices U, I, and F are often referred to as the principal component in the respective tensor mode. And the unknown affinity among user i, item j and feature k can therefore be predicted by,

$$\widehat{X}_{ijk} = \sum_{r=1}^{a}\sum_{t=1}^{b}\sum_{v=1}^{c} g_{rtv}\mathbf{u}_{ri}\mathbf{i}_{tj}\mathbf{f}_{vk}. \tag{2.1.3}$$

The predicted user's feature-level assessment can already serve as a form of rating-based explanation [78]. In the next section, we will enhance our explanation to free text based, by learning from user-provided opinionated content about the items and features.

We should note recommendation is essentially a ranking problem, in which one needs to differentiate the relative relevance quality among a set of recommendation candidates. However, the current Tucker decomposition is performed solely by minimizing element-wise reconstruction error, i.e., in Eq Eq (2.1.2), which cannot directly optimize any ranking loss. To address this limitation, we introduce the Bayesian Personalized Ranking (BPR) principle [88] into our factorization of $\widetilde{X}$. Because we only have explicit user assessments at the item-level, we introduce the BPR principle in the overall rating predictions. In particular, for each user u_i we construct a pairwise order set D_i^S based on the observations about him/her in $\widetilde{X}$:

$$D_i^S := \big\{(j,l)|\, \widetilde{x}_{ij(p+1)} > \widetilde{x}_{il(p+1)}\big\}$$

where $\widetilde{x}_{ij(p+1)} > \widetilde{x}_{il(p+1)}$ indicates in the given review set R_i^U: 1) the user i gives a higher overall rating to item j than item l; or 2) item j is reviewed by user i while item l is not. Then the BPR principle can be realized by:

$$BPR\text{-}O_{PT} := -\lambda_{\mathcal{B}} \sum_{i=1}^{m} \sum_{(j,l)\in D_i^S} \ln \sigma\big(\widehat{x}_{ij(p+1)} - \widehat{x}_{il(p+1)}\big) \tag{2.1.4}$$

in which $\lambda_{\mathcal{B}}$ is a trade-off parameter and $\sigma(\cdot)$ is the logistic function. Intuitively, Eq Eq (2.1.4) is minimized when all the pairwise ranking orders are maintained and the difference is maximized. By introducing it into the objective function of Eq Eq (2.1.2), the decomposition is forced to not only reduce element-wise reconstruction error in $\widetilde{X}$, but also to confine with the pairwise ranking order between items.

Although we only impose ranking loss over the overall rating predictions in Eq Eq (2.1.4), it also implicitly regularizes the feature-level predictions. To better understand this benefit, we can rewrite Eq Eq (2.1.3) into a matrix product form,

$$\widehat{X}_{ijk} = \mathcal{G} \times_a U_i \times_b I_j \times_c \widetilde{F}_k \tag{2.1.5}$$

where $\mathcal{G} \times_n M$ denotes the n-mode product between tensor $\mathcal{G}$ and matrix M, i.e., multiply matrix M with each mode-n fiber in $\mathcal{G}$.

For a given pair of user i and item j, the first two n-mode product results in a matrix, denoted as T_{ij}, which presents a $(p+1)\times c$ dimensional space spanned by the latent factors for user i and item j. The feature scores and overall ratings are predicted by projecting the feature factors, i.e., matrix $\widetilde{F}$, onto it. To satisfy the BRP principle in Eq Eq (2.1.4), T_{ij} has to be adjusted for each pair in D_i^S. As $\widetilde{F}$ is globally shared across users and items, this introduces the pairwise ranking loss into the gradient for all features' latent factor learning; this effect becomes more evident when we introduce the learning procedures for our joint factorization later in the Section 2.1.2.

Opinionated Content Modeling for Explanation

If an algorithm could predict the opinionated content that the user would provide on the recommended item, it is an informative explanation to reveal why the user should pay attention to those features of the recommendation. Based on this principle, we develop a companion learning task of opinionated content modeling to generate detailed textual explanations for the recommendations.

With the factorization scheme discussed in the last section, a straightforward solution for content modeling is to create a four-way tensor to summarize the complex relations among users, items, features, and opinion phrases. However, this four-way tensor would be extremely sparse in practice, as an ordinary user would only comment on a handful of items and we cannot expect their comments to be exhaustive. It is known that in natural language the distribution of words is highly skewed, e.g., Zipf's law [89]; we hypothesize that the distribution of opinion phrases that an item often receives for describing its features, and that a user often uses to describe a type of items' features are also highly skewed. In other words, the appearance of an opinion phrase towards a feature should strongly depend on the user or the item. Therefore, we approximate the four-way tensor by two three-way tensors, one summarizes the relation among user, feature and opinion phrase, and another for item, feature and opinion phrase.

This approximation is also supported by prior studies in mining opinionated text data. Amarouche [90] specifies that opinion phrase associated with a feature is apparently dependent on the opinion holder (user) as well as the target object (item) in product opinion mining. Kim and Hovy [91] focus on the importance of the opinion holder, explaining that the opinion holder's identification can be used independently to answer several opinion questions. Ronen and Moshe [92] compare products on their features/attributes by mining user-generated opinions, and report the dependence of opinions on different products features/attributes. In our experiments, we also empirically confirmed our hypothesis for approximation via a permutation test on two large review data sets.

We denote the first tensor as $Y^U \in \mathbb{R}_+^{m\times p\times q}$. From the review set R_i^U of user i, we extract all positive phrases this user has used to describe feature k across all items, i.e., $\mathcal{R}_{i,k}^U = \{o|\,(f,o,s) \in R_i^U, f = k, s = +1\}$. We only include positive phrases, as we need to explain why a user should appreciate the feature of a recommended item, rather than avoid it; otherwise we should not recommend this item or feature at all. But our algorithm can be easily extended to the scenario where one needs to provide warning messages (e.g., include the negative phrases in the tensor). To reflect the frequency of user i uses opinion phrase o to describe feature k, and to facilitate the joint factorization later, we construct Y^U as,

$$Y_{ikw}^U = \begin{cases} 0, & \text{if } w \text{ is not in } \mathcal{R}_{i,k}^U \\ 1 + (N-1)\big(\frac{2}{1+\exp(-\Gamma)} - 1\big), & \text{otherwise} \end{cases} \tag{2.1.6}$$

where Γ is the frequency of phrase w in $\mathcal{R}_{i,k}^U$.

We construct the second tensor $Y^I \in \mathbb{R}_+^{n\times p\times q}$ in a similar way. For item j, we first obtain a collection of positive phrases about its feature k from R_j^I, i.e., $\mathcal{R}_{j,k}^I = \{o|(f,o,s) \in R_j^I, f = k, s = +1\}$, and then construct Y^I as:

$$Y_{jkw}^I = \begin{cases} 0, & \text{if } w \text{ is not in } \mathcal{R}_{j,k}^I \\ 1 + (N-1)\big(\frac{2}{1+\exp(-\Omega)} - 1\big), & \text{otherwise} \end{cases} \tag{2.1.7}$$

where Ω is the frequency of phrase w in $\mathcal{R}_{j,k}^I$.

The construction of tensor $\widetilde{X}$, Y^U and Y^I impose strong dependency between the two learning tasks of item recommendation and opinionated explanation, as every two tensors share the same two types of entities (as shown in Figure 2.2). Instead of isolating the factorization of these three tensors, we propose a joint factorization scheme, which will be discussed in detail in the next section.

Once the latent factors are learnt, we can predict user i's opinionated comments on feature k by the reconstructed vector $\widehat{Y}_{i,k}^U$, which can be calculated in the same way as in Eq Eq (2.1.3) with the corresponding latent factors. Similarly, the opinionated comments that item j will receive on its feature k can be predicted by the reconstructed vector $\widehat{Y}_{j,k}^I$. As a result, to predict the opinionated comments that user i will provide on item j's feature k, we take an element-wise product between these two vectors to construct an opinion phrase scoring vector $\widehat{Y}_{i,j,k}^{U,I}$, in which each element is computed as,

$$\widehat{Y}_{i,j,k,w}^{U,I} = \widehat{Y}_{i,k,w}^U \times \widehat{Y}_{j,k,w}^I \tag{2.1.8}$$

This estimation reflects our approximation of the original four-way tensor with two three-way tensors. Because the tensor Y^U and Y^I record the frequency of an opinion phrase used in describing the features by the user

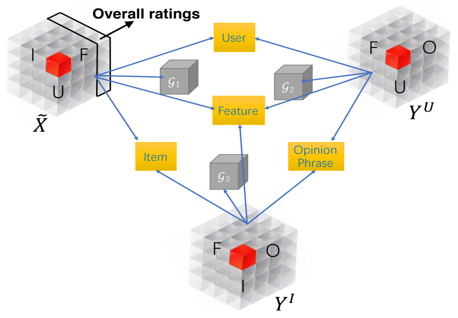


Figure 2.2: Joint tensor decomposition scheme. Task relatedness is captured by sharing latent factors across the tensors; in-task variance is captured by corresponding core tensors.

and about the item, Eq (2.1.8) prefers to choose those that are popularly used to describe this feature of the item in general, and also by this target user to describe this feature in similar items.

Multi-task Learning via a Joint Tensor Factorization

Both of our proposed learning tasks are modeled as a tensor factorization problem, and they are coupled with the shared latent factors. Ideally, the predicted users' assessment about the recommendation candidates from the first task should be supported by the predicted users' opinionated comments about the recommendations from the second task. To leverage the dependency between these two tasks, we develop a joint factorization scheme.

In Tucker decomposition, a three-way input tensor will be decomposed into a core tensor and three principle component matrices. The core tensor captures multivariate interactions among the latent factors; and the principle component matrices can be viewed as basis of the resulting latent space. Based on this property, we decide to share the principle component matrices across the three tensors of $\widetilde{X}$, Y^U and Y^I to learn the latent representations of user, item, feature and opinion phrases across the two learning tasks, and keep independent core tensors for these tensors to capture the tasks' intrinsic variance and scaling of the shared latent factors. As a result, we devise the following joint optimization formulation,

$$\begin{aligned}
\min_{\widehat{X},\widehat{Y}^U,\widehat{Y}^I} & ||\widehat{X}-\widetilde{X}||_F + ||\widehat{Y}^U - Y^U||_F + ||\widehat{Y}^I - Y^I||_F - \lambda_{\mathcal{B}}\sum_{i=1}^{m}\sum_{(j,l)\in D_i^S}\ln\sigma\big(\widehat{x}_{ij(p+1)} - \widehat{x}_{il(p+1)}\big) \\
& + \lambda_{\mathcal{F}}\big(||U||^2 + ||I||^2 + ||F||^2 + ||O||^2\big) + \lambda_{\mathcal{G}}\big(||\mathcal{G}_1||^2 + ||\mathcal{G}_2||^2 + ||\mathcal{G}_3||^2\big) \\
\text{s.t.}\quad & \widehat{X} = \mathcal{G}_1 \times_a U \times_b I \times_c \widetilde{F}, \\
& \widehat{Y}^U = \mathcal{G}_2 \times_a U \times_c F \times_d O, \\
& \widehat{Y}^I = \mathcal{G}_3 \times_b I \times_c F \times_d O, \\
& U \geq 0, I \geq 0, F \geq 0, O \geq 0, \mathcal{G}_1 \geq 0, \mathcal{G}_2 \geq 0, \mathcal{G}_3 \geq 0. \qquad (2.1.9)
\end{aligned}$$

where we introduce l_2 regularization over the learned latent factor matrices and core tensors to avoid over-fitting. This joint factorization ensembles the two companion learning tasks for recommendation and explanation, i.e., multi-task learning; and therefore, we name our solution as Multi-Task Explainable Recommendation, or MTER in short.

The above optimization problem can be effectively solved by stochastic gradient descent (SGD), with projected gradients for non-negative constraints. However, because the number of observations in each tensor and in the pairwise ranking constraint set varies significantly, vanilla SGD procedure suffers from local optimum. To improve the convergence, we randomly select small batches of samples from each tensor and pairwise constraint set per iteration to calculate a averaged gradient, i.e., a mini-batch SGD. And to avoid manually specifying a step size, we employ adaptive gradient descent [93], which dynamically incorporates the updating trace in earlier iterations to perform more informative and faster gradient-based learning. The parameter

Table 2.1: Basic statistics of evaluation datasets.

Dataset	#users	#items	#features	#opinions	#reviews
Amazon	6,285	12,626	95	591	55,388
Yelp	10,719	10,410	104	1,019	285,346

estimation procedure of our model is off-line, and large-scale learning tasks could be solved within reasonable periods.

Interactions between the two learning tasks become more evident in MTER when we look into the detailed gradients for model update. Denote the objective function in Eq Eq (2.1.9) as L, and we list the gradient of F_k as an example to illustrate how the elements in three tensors $\widetilde{X}$, Y^U and Y^I contribute to it:

$$\frac{\partial L}{\partial_{F_k}} = \frac{\partial L}{\partial_{\widehat{X}_{ijk}}} \mathcal{G}_1 \times_a U_i \times_b I_j + \frac{\partial L}{\partial_{\widehat{Y}^U_{ikl}}} \mathcal{G}_2 \times_a U_i \times_d O_w + \frac{\partial L}{\partial_{\widehat{Y}^I_{jkl}}} \mathcal{G}_3 \times_b I_j \times_d O_w \tag{2.1.10}$$

In Eq Eq (2.1.10), as F_k is shared across the decomposition of all three tensors, it bridges the other three components U_i, I_j and O_w in these two tasks. Similarly, the gradient of U_i, I_j and O_w also involves all the rest factors. Furthermore, the BPR constraint introduced on overall rating prediction indirectly affects the learning of U_i, I_j and O_w, via gradient sharing. This also helps MTER conquer data sparsity issue when we have a large number of users, items, features and opinionated phrases to model.

2.1.3 Experiments

In this section, we quantitatively evaluate our solution MTER in the tasks of item recommendation and opinionated content modeling, on two popular benchmark datasets collected from Amazon[1] [94,95] and Yelp Dataset Challenge[2]. We perform extensive comparisons against several state-of-the-art recommendation and explainable recommendation algorithms. Improved quality in both recommendation and opinionated content prediction confirms the comprehensiveness and effectiveness of our solution.

Experiment Setup

Datasets and Preprocessing. To verify our model's effectiveness in different application domains, we choose *restaurant* businesses from Yelp dataset and *cellphones and accessories* category from Amazon dataset. These two datasets are very sparse: 73% users and 47% products only have one review in Amazon dataset, and 54% users only have one review in Yelp dataset. However, in the constructed sentiment lexicons, 401 features are extracted from Amazon dataset, and 1065 are extracted from Yelp dataset. It is very difficult to estimate the affinity between users and those hundreds of features from only a handful of reviews. To refine the raw datasets, we first analyze the coverage of different features in these two datasets. Within the sentiment lexicon, only a small number of features (around 15%) that are frequently covered in 90% reviews, while most of features occur rarely in the whole datasets (i.e., Zipf's law). As a result, we perform recursive filtering to alleviate the sparsity issue. First, we select features whose support is above a threshold; then, in turn, we filter out reviews that mentions such features below another threshold, and items and users that are associated with too few reviews. By fine tuning these different thresholds, we obtain two refined datasets with decent amount of users and items, whose basic statistics are reported in Table 2.1.

Baselines. To evaluate the effectiveness of our proposed explainable recommendation solution, we include the following recommendation algorithms as baselines:

- **FMF:** Functional Matrix Factorization [96]. It constructs a decision tree on the user side for tree-based matrix factorization. It was originally designed to solicit interview questions on each tree node for cold-start recommendations.

[1] http://jmcauley.ucsd.edu/data/amazon/
[2] https://www.yelp.com/dataset

Table 2.2: Comparison of recommendation performance.

NDCG @K	Amazon						
	FMF [96]	NMF [97]	BPRMF [98]	JMARS [85]	EFM [78]	MTER	FacT
10	0.1009	0.0649	0.1185	0.1064	0.1109	**0.1351**	**0.1482**
20	0.1331	0.0877	0.1490	0.1348	0.1464	**0.1653**	**0.1795**
50	0.1976	0.1601	0.2070	0.1992	0.2056	**0.2234**	**0.2367**
100	0.2529	0.2144	0.2669	0.2575	0.2772	**0.2803**	**0.2869**
NDCG@K	Yelp						
	FMF [96]	NMF [97]	BPRMF [98]	JMARS [85]	EFM [78]	MTER	FacT
10	0.0931	0.0564	0.1266	0.1155	0.1071	**0.1380**	**0.1499**
20	0.1243	0.0825	0.1643	0.1553	0.1354	**0.1825**	**0.1991**
50	0.1871	0.1345	0.2214	0.2111	0.1903	**0.2365**	**0.2488**
100	0.2509	0.2175	0.2668	0.2575	0.2674	**0.2783**	**0.2867**

* p-value < 0.05

- **NMF:** Nonnegative Matrix Factorization [97], which is a widely applied latent factor model for recommendation.
- **BPRMF:** Bayesian Personalized Ranking on Matrix Factorization [88], which introduces BPR pairwise ranking constraint into factorization model learning (as shown in Eq Eq (2.1.4)).
- **JMARS:** A probabilistic model that jointly models aspects, ratings, and sentiments by collaborative filtering and topic modeling [85].
- **EFM:** Explicit Factor Models [78]. A joint matrix factorization model for explainable recommendation, which considers user-feature attention and item-feature quality.

Evaluation Metric. We use Normalized Discounted Cumulative Gain (NDCG) to evaluate top-k recommendation performance. 80% of each dataset is used for training, 10% for validation and 10% for testing respectively. We use grid search to find the optimal hyper parameters in a candidate set for all baseline models.

Experiment Results

Performance of Recommendation. We report the recommendation performance of each model measured by NDCG@{10,20,50,100} in Table 2.2. Paired t-test is performed between the best and second best performing algorithms (FacT excluded which will be introduced in the next section) under each metric to confirm the significance of improvement.

Results in Table 2.2 demonstrate the advantage of MTER over the baselines. Straightforward factorization algorithm (i.e., NMF) cannot optimize the ranking quality of the recommended items. The pairwise ranking constraints introduced by BPR greatly improve the recommendation effectiveness of BPRMF, which shares the same decomposition structure as in NMF. However, as BPRMF only models users' overall assessment on items, it cannot exploit information available in the user-provided opinionated content. Second, comparing to JMARS and EFM, which also utilize review content for recommendation, MTER is the only model that outperforms BPRMF. JMARS models all entities in a shared topic space, which limits it resolution in modeling complex dependencies, such as users v.s., items, and users v.s., features. EFM implicitly integrates the interaction among users, items and features via three loosely coupled matrices, and it is only optimized by the reconstruction error on those three matrices. This greatly limits its recommendation quality. We can also observe that the best improvement from MTER is achieved at NDCG@10 (more than 15% against the best baseline on Amazon and over 11% on Yelp). This result is significant: it indicates a system equipped with MTER can provide satisfactory results earlier down the ranked list, which is crucial in all practical recommender systems.

Opinionated Textual Explanation. We study the effectiveness of our opinionated content modeling task by evaluating if our model can predict the actual review content a user would provide on a testing item. In particular, we focus on whether our model can recognize: 1) the features that the user would pay attention to in a given item, and 2) the detailed opinion phrases the user would use to describe this particular feature. The model has to leverage observations across different users, items and features, to infer their dependency. In our experiment, we use the learnt latent factors to score all possible features associated with a given item in

Table 2.3: Performance of opinionated content prediction.

		Random	EFM	MTER
Feature Pred.	Amazon	0.4843	0.5094	**0.5176***
NDCG@20	Yelp	0.3726	0.3929	**0.4089***
Opinion Phrase Pred.	Amazon	0.0314	-	**0.0399***
NDCG@50	Yelp	0.0209	-	**0.0370***

*p-value < 0.05

each user, and look into the user's review to verify if the top ranked features are indeed mentioned. Similarly, we also evaluate the ranking of all possible opinion phrases associated with a specific feature to test if our model can put the users' choice on top. As EFM is the only baseline that predicts feature-level opinions, we include it as our baseline for comparison. However, because EFM cannot predict detailed opinion phrases, we also use a simple random strategy based on the feature popularity and opinion phase popularity in target user and item as our baseline. We report the results in Table 2.3. MTER achieves promising performance in ranking the features that a user will mention; this proves it identifies users' true feature-level preference, which is important for both recommendation and explanation. In the opinion phrase prediction, although it is a very difficult task, MTER is still able to predict the detailed reasons that a user might endorse the item. This indirectly confirms MTER's effectiveness in explaining the recommendation results, which will be directly evaluated in our user study later.

2.2 Explainable Recommendation with Factorization Trees

MTER incorporates the phrase-level sentiment analysis into latent factor learning, e.g., joint tensor factorization, for explanations. Basically, MTER maps users' feature-level opinions into the latent space and finds the most related features to the users and recommended items as explanations. However, to what extent these approximated explanations comply with the learned latent factor models is unknown, i.e., *no guarantee in explanation fidelity*. If users are persuaded to accept recommended results that are subsequently found to be inferior, their confidence and trust in the system will rapidly deteriorate [14, 35]. Hence, the fidelity of explanations becomes a prerequisite for explainable recommendations.

Nevertheless, the fidelity of explanation and the quality of recommendation have long been considered as irreconcilable [11]: one has to trade recommendation quality for explanation. We believe the tension between recommendation quality and explanation fidelity is not necessarily inevitable; and our goal is to attain both by optimizing the recommendation in accordance with the designed explanation mechanism. In this work, we aim at explaining latent factor based recommendation algorithms with rule-based explanations. Our choice is based on the facts that 1) latent factor models have proved their effectiveness in numerous practical deployments [28, 99], and 2) prior studies show that rule-based explanations are easy to perceive and justify by the end-users [36]. We propose to integrate the rule-based decision making into the learning of latent factors. More specifically, we treat the latent factors as a function of the rules: based on different outcome of the rules, the associated groups of users and items should be routed to the designated latent factors, which are then optimized for recommendation. Due to similar characteristics shared by each group of users/items created by the learnt rules, the descriptive power of the learnt *group-level* latent factors is enhanced, and the data sparsity problem in individual users/items could be substantially alleviated by this group-level latent factor learning.

More specifically, we format the explanation rules based on feature-level opinions extracted from user-generated review content, e.g., whether a user holds positive opinion towards a specific feature. The rules are extracted by inductive learning on the user side and item side separately, which form a user tree and an item tree. We alternate the optimization between tree construction and latent factor estimation under a shared recommendation quality metric. An example of user tree is shown in Figure 2.3. For instance, according to the figure, if two users both expressed their preference of "*burger*" in their reviews, they should be assigned to the same node on the user tree to share the latent user factors; accordingly, if two restaurants receive similar negative comments about their "*cleanliness*", they should appear in the same node on the item tree. In testing time, the learned user and item factors are used for recommendation as in standard latent factor models, and the rules that lead to the chosen user and item factors are output as explanations: e.g., "*We recommend item X because it matches your preference on burger and cleanness of a restaurant.*" Extensive experiment evaluations demonstrate improved quality in recommendation and explanation from our algorithm, compared with a set

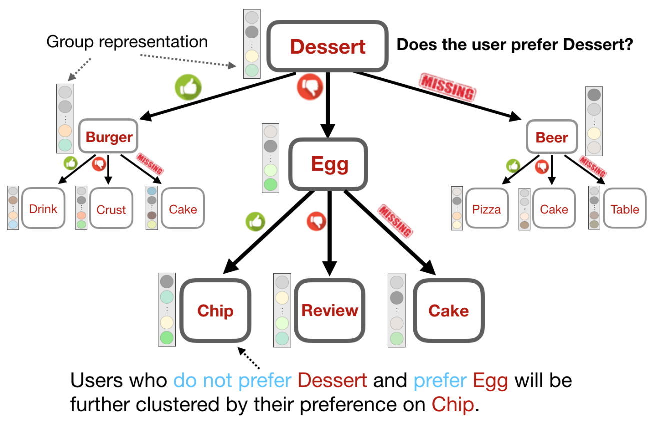

Group representation
Dessert
Does the user prefer Dessert?
MISSING
Burger
Egg
Beer
Drink
Crust
Cake
Pizza
Cake
Table
Chip
Review
Cake
Users who do not prefer Dessert and prefer Egg will be further clustered by their preference on Chip.

Figure 2.3: An example user tree: Top three levels of our FacT model learned for restaurant recommendations.

of state-of-the-art explainable recommendation algorithms. In particular, we perform serious user studies to investigate the utility of our explainable recommendation in practice, in both warm-start and cold-start settings. Positive user feedback further validates the value of our proposed solution.

2.2.1 Related Work

The idea of providing rule-based explanations was popularized in the development of expert systems [100, 101]. For example, MYCIN [102], a rule-based reasoning system, provides explanations by translating traces of rules followed from LISP to English. A user could ask both why a conclusion was arrived at and how much was known about a certain concept. But since modern recommender systems seldom use rule-based reasoning, there is very little research on explaining latent factor models with rules. We propose to embed latent factor learning under explanation rule learning, by treating the latent factors as a function of rules, such that the generated explanations can strictly adhere to the provided recommendations. On a related note, a existing work [103] uses gradient boosting decision trees (GBDT) to learn rules from the reviews and incorporate rules into an attention network. But it only uses the rules as the input of embedding models and thus isolates the learning of tree structure and embeddings. Some systems [96, 104] combine decision tree learning with matrix factorization to extract a list of interview questions for solving the cold-start problem in recommendation. But the rules are only built on the user side with their rating responses to items, i.e., the same as matrix factorization's input; it thus cannot provide any explanation to the recommended items.

2.2.2 Taming Latent Factor Models with Factorization Tree

We elaborate our solution for joint latent factor learning and explanation rule construction in this section. Briefly, we model the latent factors for both users and items as a function of the rules: users who provide the same responses to the same set of rules would share the same latent factors, and so do the items. The predicates of rules are selected among the text features extracted from user-generated reviews. For example, whether a specific user expressed his/her preference on a particular feature in reviews. And the rules are constructed by recursive inductive learning based on previously selected predicate's partition of users and items. To reflect the heterogeneity between users and items, we construct rules for users and items separately. As a result of rule induction, the latent factors for users and items are organized in a decision tree like structure accordingly, where each node on the tree represents the latent factors for the group of users or items routed to that node. We alternate the optimization of the explanation rule construction and latent factor learning under a recommendation quality based metric. Hence, we name our solution as *Factorization Tree*, or *FacT* in short.

We start our discussion with factorization based latent factor learning, which is the basic building block of FacT. Then we provide details in rule induction based on the learned latent factors. Finally, we integrate these two learning components with an alternative optimization procedure.

Latent Factor Learning

Latent factor models [28,99] have been widely deployed in modern recommender systems. The idea behind this family of solutions is to find vectorized representations of users and items in a lower dimensional space, which capture the affinity between users and items. Various latent factor models have been developed, such as matrix/tensor factorization [99] and factorization machines [28]. Our FacT is independent of the choice of latent factor models, as it treats the latent factor learning as a sub-routine.

Formally, denote $\mathcal{U} = \{u_1, u_2, ..., u_m\}$ as a set of m users, $\mathcal{V} = \{v_1, v_2, ..., v_n\}$ as a set of n items, and r_{ij} as an observed rating from user i to item j. The goal of latent factor learning is to associate each user and each item an d dimensional vector respectively, i.e., $u_i \in \mathbb{R}^d$ and $v_j \in \mathbb{R}^d$, such that the inner product between user i's factor and item j's factor predicts the rating r_{ij}. The latent factors for all the users and items, denote as $U \in \mathbb{R}^{m\times d}$ and $V \in \mathbb{R}^{n\times d}$, can thus be learned by minimizing their prediction error over a set of observed ratings $O = \{(i,j)|r_{ij}\, is\, observed\}$ as follows,

$$\mathcal{L}(U,V,O) = \min_{U,V} \sum_{(i,j)\in O} (r_{ij} - u_i^\top v_j)^2. \tag{2.2.1}$$

It is well accepted that recommendation is essentially a ranking problem [98, 105]; however, the objective function introduced in Eq Eq (2.2.1) cannot fully characterize the need of ranking, i.e., differentiate the relative quality among candidates. To supplement information about relative item ranking into latent factor learning, Bayesian Personalized Ranking (BPR) loss [98] has been popularly adopted to enforce pairwise ranking order. To realize the BPR loss, one needs to first construct a pairwise ordered set of items D_i^o for each user i: $D_i^o := \{(j,l)|\, r_{ij} > r_{il}\}$, where $r_{ij} > r_{il}$ means that given the observations in O, either user i gives a higher rating to item j than item l, or item j is observed in user i's rating history, while item l is not. Then, the BPR loss can be measured on each user i as:

$$\mathcal{B}(u_i, V, D_i^o) = \sum_{(j,l)\in D_i^o} \log\sigma(u_i^\top v_j - u_i^\top v_l)$$

where $\sigma(\cdot)$ is a logistic function.

Putting together the pointwise rating prediction loss with the pairwise ranking loss, the latent factors for users and items can be learned by solving the following optimization problem:

$$(\widehat{U}, \widehat{V}) = \operatorname*{argmin}_{U,V} \mathcal{L}(U,V,O) - \lambda_b \sum_i \mathcal{B}(u_i, V, D_i^o) + \lambda_u \|U\|_2 + \lambda_v \|V\|_2 \tag{2.2.2}$$

where λ_b is a trade-off parameter to balance these two types of loss, $\|U\|_2$ and $\|V\|_2$ are l_2 regularization to control model complexity, and λ_u and λ_v are the corresponding coefficients. Eq Eq (2.2.2) can be efficiently addressed by gradient-based optimization [106]. Once the user factors U and item factors V have been learned, the recommendations for user i can be generated by returning the top ranked items based on the predicted ratings $\widehat{r}_{ij} = \widehat{u}_i^\top \widehat{v}_j$.

The premise behind the aforementioned learning procedure is that there is only a small number of factors influencing users' preferences, and that a user's preference vector is determined by how each factor applies to that user and associated items. But the factors are retrieved by solving a complex optimization problem (e.g., Eq Eq (2.2.2)), which makes the resulting recommendations hard to explain. In FacT, we embed latent factor learning under explanation rule construction, so that why a user or an item is associated to a particular latent factor can be answered by the matched rules, so do the generated recommendations.

Explanation Rule Induction

In FacT, we consider the latent factors as a function of explanation rules: the latent user factor u_i for user i is tied to the outcomes of a set of predicates applied to him/her, so does the latent item factor v_j for item j. Based on different outcomes of the rules, the associated groups of users and items should be routed to the designated latent factors. At testing time, the activated predicates on user i and item j naturally become the explanation of this recommendation.

We select the predicates among the item features extracted from user-generated reviews. User reviews provide a fine-grained understanding of a user's evaluation of an item. Feature-level sentiment analysis techniques [33] can be readily applied to reviews to construct a domain-specific sentiment lexicon. Each lexicon entry takes the form of (feature, opinion, sentiment polarity), abbreviated as (f, o, s), and represents the sentiment polarity s inferred from an opinionated text phrase o describing feature f. Specifically, we label the sentiment polarity s as +1 or -1, to represent positive or negative opinions. As how to construct a sentiment lexicon with phrase-level sentiment analysis is not the focus of this work, we refer interested readers to [12, 33] for more details.

The extracted item features become candidate variables for predicate selection. To compose predicates for explanation rule construction, we first need to define the evaluation of a single variable predicate on users/items according to their association with the item features. To respect the heterogeneity between users and items, we construct the predicates for users and items separately; but the construction procedures are very similar and highly related on these two sides.

Denote $\mathcal{F} = \{f_1, f_2, ..., f_k\}$ as a set of k extracted item features. Suppose feature f_l is mentioned by user i for p^u_{il} times with a positive sentiment polarity in his/her reviews and n^u_{il} times with a negative sentiment polarity. We can construct a feature-level profile F^u_i for user i, where each element of F^u_i is defined as,

$$F^u_{il} = \begin{cases} \emptyset, & \text{if} \quad p^u_{il} = n^u_{il} = 0, \\ p^u_{il} + n^u_{il}, & \text{otherwise.} \end{cases} \tag{2.2.3}$$

Intuitively, F^u_{il} is the frequency of user i mentioning feature f_l in his/her reviews, such that it captures the relative emphasis that he/she has given to this feature. And similarly, on the item side, denote p^v_{jl} as the number of times that feature f_l is mentioned in all user-generated reviews about item j with a positive sentiment polarity, and n^v_{jl} as that with a negative sentiment polarity, we define the feature-level profile F^v_j for item j as,

$$F^v_{jl} = \begin{cases} \emptyset, & \text{if} \quad p^v_{jl} = n^v_{jl} = 0, \\ p^v_{jl} - n^v_{jl}, & \text{otherwise.} \end{cases} \tag{2.2.4}$$

Accordingly, F^v_{jl} reflects the aggregated user sentiment evaluation about feature f_l of item j.

Based on the feature-level user and item profiles, the evaluation of a single variable predicate can be easily performed by comparing the designated feature dimension in the user or item profile against a predefined threshold. For example, on the user side, if a predicate is instantiated with feature f_l and threshold t^u_l, all users can have three disjoint responses to this predicate based on their F^u_{il} values, i.e., $F^u_{il} \geq t^u_l$, or $F^u_{il} < t^u_l$, or F^u_{il} is unknown. This gives us the opportunity to model the latent factors as a function of the explanation rules: based on the evaluation results of a predicate, we allocate the input users into three separate user groups and assign one latent vector per group. We should note that other forms of predicates are also applicable for our purpose, e.g., select a list of thresholds or a nonlinear function for one variable. For simplicity, we adhere to the form of single threshold predicates, and leave the more complex forms of predicates for future exploration.

Two questions remain to be answered: First, how to select the threshold for user-side and item-side predicate creation; and second, how to assign latent vectors for each resulting user/item group. We answer the first question in this section by inductive rule learning, and leave the second to the next section, where we present an alternative optimization procedure for joint rule learning and latent factor learning. In the following discussions, we will use user-side predicate construction as an example to illustrate our rule induction method; and the same procedure directly applies to item-side predicate construction.

Intuitively, an optimal predicate should create a partition of input users where the latent factors assigned to each resulting user group lead to minimal recommendation loss defined in Eq Eq (2.2.2). This can be achieved by exhaustively searching through the combination of all item features in $\mathcal{F}$ and all possible corresponding thresholds. This seems infeasible at a first glance, as the combinatorial search space is expected to be large. But in practice, due the sparsity of nature language (e.g., Zipf's law [107]), the mentioning of item features and its frequency in user reviews are highly concentrated at both user-level and item-level [108]. Besides, feature discretization techniques [109] can also be used to further reduce the search space.

To perform the search for optimal predicate in an input set of users $\mathcal{U}_a$, we first denote the resulting partitions of $\mathcal{U}_a$ by feature f_l and threshold t_l^u as,

$$\begin{aligned} L(f_l, t_l^u|\mathcal{U}_a) &= \{i|F_{il}^u \geq t_l^u, i \in \mathcal{U}_a\}, \\ R(f_l, t_l^u|\mathcal{U}_a) &= \{i|F_{il}^u < t_l^u, i \in \mathcal{U}_a\}, \\ E(f_l, t_l^u|\mathcal{U}_a) &= \{i|F_{il}^u = \emptyset, i \in \mathcal{U}_a\}, \end{aligned} \tag{2.2.5}$$

and the set of possible threshold t_l^u for feature f_l as T_l^u. The optimal predicate on $\mathcal{U}_a$ can then be obtained by solving the following optimization problem with respect to a given set of item factors V,

$$\begin{aligned} (\bar{f}_l, \bar{t}_l^u) = \underset{f_l \in \mathcal{F}, t_l^u \in T_l^u}{\operatorname{argmin}} \min_{u_L, u_R, u_E} & \mathcal{L}(u_L, V, O_L) - \lambda_b \sum_{i \in E(f_l, t_l^u)} \mathcal{B}(u_L, V, D_i^o) \\ & + \mathcal{L}(u_R, V, O_R) - \lambda_b \sum_{i \in R(f_l, t_l^u)} \mathcal{B}(u_R, V, D_i^o) \\ & + \mathcal{L}(u_E, V, O_E) - \lambda_b \sum_{i \in E(f_l, t_l^u)} \mathcal{B}(u_E, V, D_i^o) \\ & + \lambda_u(\|u_L\|_2 + \|u_R\|_2 + \|u_E\|_2) \end{aligned} \tag{2.2.6}$$

where O_L, O_R and O_E are the observed ratings in the resulting three partitions of $\mathcal{U}_a$, and u_L, u_R and u_E are the correspondingly assigned latent factors for the users in each of the three partitions. As users in the same partition are forced to share the same latent factors, the choice of text feature f_l and corresponding threshold t_l^u directly affect recommendation quality. In practice, considering each user and item might associate with different number of reviews, the size of user profile F_i^u and item profile F_j^v might vary significantly. Proper normalization of F_i^u and F_j^v can be performed, e.g., normalize by the total observation of feature mentioning in each user and item respectively. In this work, we follow [109] for feature value normalization and discretization.

Inside the optimization of Eq Eq (2.2.6), a sub-routine of latent factor learning is performed to minimize recommendation loss induced by matrix factorization (as defined in Eq Eq (2.2.2)) on the resulting partition of users. As we mentioned before, the choice of latent factor models does not affect the procedure of our predicate construction for FacT, and many other recommendation loss metrics or latent factor models can be directly plugged into Eq Eq (2.2.2) for explainablity enhancement. We leave this exploration as our future work.

Our predicate construction can be recursively applied on the resulting user partitions $L(\bar{f}_l, \bar{t}_l^u|\mathcal{U}_a)$, $R(\bar{f}_l, \bar{t}_l^u|\mathcal{U}_a)$ and $E(\bar{f}_l, \bar{t}_l^u|\mathcal{U}_a)$ on the input user set $\mathcal{U}_a$ to extend a single variable predicate to a multi-variable one, i.e., inductive rule learning. The procedure will be terminated when, 1) the input user set cannot be further separated, e.g., all users there share the same user profile; or 2) the maximum depth has been reached. Starting the procedure from the complete set of users $\mathcal{U}$, the resulting set of multi-variable predicates form a decision tree like structure, which we refer as user tree in FacT (as shown in Figure 2.3). On the user tree, each node hosts a latent factor assigned to all its associated users, and its path to the root node presents the learned predicates for this node. The same procedure can be applied on the item side with a given set of user factors U to construct item-specific predicates, i.e., item tree.

Once the user tree and item have been constructed, explaining the recommendations generated by the latent factors becomes straightforward. Assume we recommend item j to user i: we first locate user i and item j at the leaf nodes of user tree and item tree accordingly, extract their paths back to each tree's root node, and find the shared features on the paths to create feature-level explanations. As each branch on the selected path corresponds to a specific outcome of predicate evaluation, e.g., Eq Eq (2.2.5), we can add predefined modifiers in front of the selected features to further elaborate the associated latent factors. For example,

- *We recommend this item to you because its [good/excellent] [feature 1] matches with your [emphasize/taste] on [feature 1], and ...*
- *We guess you would like this item because of your [preference/choice] on [feature 1], and ...*

It is also possible that the number of shared features on the two paths is low, especially when the maximum tree depth is small. In this situation, one can consider to use the union of features on these two paths, and give higher priority to the shared features and those at the lower level of the trees, as they are more specific. Another possible way of explanation generation is to use the selected features to retrieve sentences from the corresponding item reviews [110].

Alternative Optimization

The aforementioned procedure for explanation rule induction is intrinsically recursive and requires the availability of user factors for item tree construction and item factors for user tree construction. In this section, we will unify the learning of latent factors with tree construction to complete our discussion of FacT.

Define the maximum rule length, i.e., tree depth, as h. We alternate rule induction by recursively optimizing Eq Eq (2.2.6) between user side and item side. At iteration t, we start induction from the complete user set $\mathcal{U}$ with the latest item factors V_{t-1}. For each pair of feature and threshold in the hypothesis space of Eq Eq (2.2.6), we use gradient based optimization for learning latent factors according to Eq Eq (2.2.2). Once the induction finishes, we collect the latent user factors U_t from the leaf nodes of the resulting user tree, and use them to execute the rule induction on the item side from the complete item set $\mathcal{V}$ to estimate V_t. This procedure is repeated until the relative change defined in Eq Eq (2.2.2) between two consecutive iterations is smaller than a threshold, or the maximum number of iterations is reached. To break the inter-dependency between item tree and user tree construction, we first perform plain matrix factorization defined in Eq Eq (2.2.2) to obtain the initial item factors V_0. We should note that one can also start with item tree construction from initial user factors U_0, but this does not change the nature and convergence of this alternative optimization.

The above alternative optimization is by nature greedy, and its computational complexity is potentially high. When examine the optimization steps in Eq Eq (2.2.6), we can easily recognize that the exhaustive search of item features and thresholds can be performed in parallel in each input set of users and items. This greatly improves the efficiency of rule induction. Besides, beam search [111] can be applied in each step of predicate selection to improve the quality of learned rules and factors, but with a cost of increased computation.

One can realize that during the alternative optimization, only the latent factors learned for the leaf nodes are kept for next round of tree construction and finally the recommendation, while the factors associated with the intermediate nodes are discarded. As the procedure of inductive rule learning can be considered as a process of divisive clustering of users and items, the intermediate nodes actually capture important information about homogeneity within the identified user clusters and item clusters. To exploit such information, we introduce the learned latent factors from parent node to child node as follows,

$$u_{L,z} = \widetilde{u}_{L,z} + u_z, \;\; u_{R,z} = \widetilde{u}_{R,z} + u_z, \;\; \text{and} \;\; u_{E,z} = \widetilde{u}_{E,z} + u_z,$$

where $u_{L,z}$, $u_{R,z}$ and $u_{E,z}$ are the latent factors to be plugged into Eq Eq (2.2.6) for the three child nodes under parent node z, and u_z is the factor already learned for the parent node z. Intuitively, $\widetilde{u}_{L,z}$, $\widetilde{u}_{L,z}$ and $\widetilde{u}_{L,z}$ can be considered as residual corrections added to the shared representation from parent nodes. Hence, the rule induction process becomes a recursive procedure of latent factor refinement. Without loss of generality, this recursive refinement can be applied to individual users and items on the leaf nodes of both user tree and item tree as well. If we refer the latent factors on the leaf node for individual users and items as personalized representations of users and items, those on the intermediate nodes could be considered as grouplized representations for the partition of users and items.

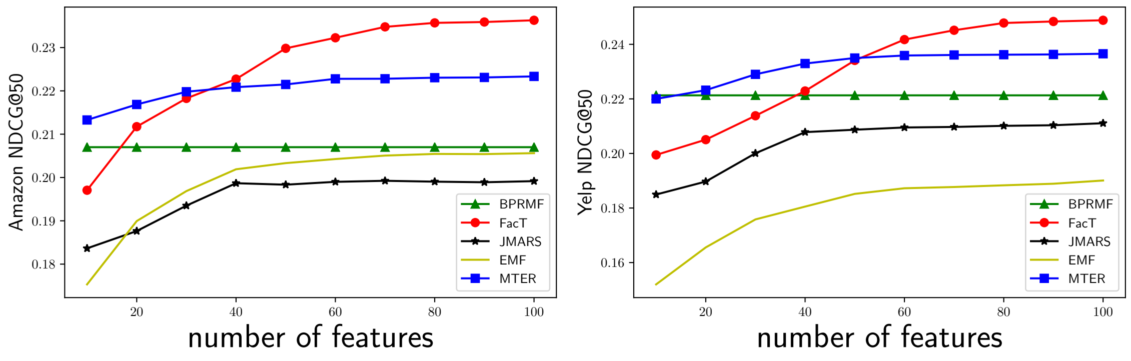


Figure 2.4: NDCG@50 v.s. the number of item features.

2.2.3 Experiments

We performed a set of controlled experiments on two widely used benchmark datasets collected from Amazon[3] and Yelp Dataset Challenge[4] to quantitatively evaluate our FacT model. We follow the same technique discussed in Section 2.1.3 to perform pre-processing on the dataset, and the detailed statistic can be found in Table 2.1. Besides, we compare FacT against the same baselines as that for MTER in both recommendation and explanation quality.

Top-K Recommendation

We first evaluate FacT's recommendation quality. In a good recommender system, items ranked higher in a result list should be more relevant to a user's preference. NDCG assigns higher importance to the items ranked on top. In this experiment, we fix the depth of the user tree and item tree in FacT to 6 and the size of latent dimension to 20. The recommendation performance measured by NDCG@10, 20, 50, 100 of each model is shown in Table 2.2 for Amazon and Yelp datasets, respectively.

Compared with all the baselines, FacT consistently gives better recommendation in both Amazon and Yelp datasets. By exploiting the review content for recommendation, JMARS and EFM gave explainable recommendation to users with comparable ranking quality with BPRMF, and MTER showed its potential in providing explanations along with decent recommendation quality. However, they are still limited for different reasons. JMARS maps users, items and features into the same topic space, where the dependency among them is not preserved. Both EMF and MTER model the users and items as individual vectors by matrix or tensor factorization, while FacT clusters users and items into groups (e.g., the intermediate nodes in user and item trees) to take advantage of in-group homogeneity for better latent factor learning. The basic intuition here is that the representations of users and items that share the same preferences or feature qualities should be pushed close to each other. And FacT enforced it by item feature based tree construction. Moreover, the personalized vectors added to the leaf nodes distinguish individual users/items, and provide accurate personalized recommendations.

Number of item features

As shown in Table 2.1, there are 101 item features extracted from Amazon dataset and 104 features from Yelp dataset. Though we have filtered out features with low frequency, limited by the depth of our tree structure, not all of these features will be selected for rule construction. In this analysis, we study the impact of number of features in the dataset on the performance of different models. We first ordered the features in a descending order of frequency, and then trained the models with an increasing number of features. The results are reported in Figure 2.4. From Figure 2.4, it is easy to observe that all the models get significantly improved with an increasing number of features. As the number of features got larger, the performance became stable, as more less frequent features were added. This observation suggests that features with high frequency in reviews

[3]http://jmcauley.ucsd.edu/data/amazon/
[4]https://www.yelp.com/dataset

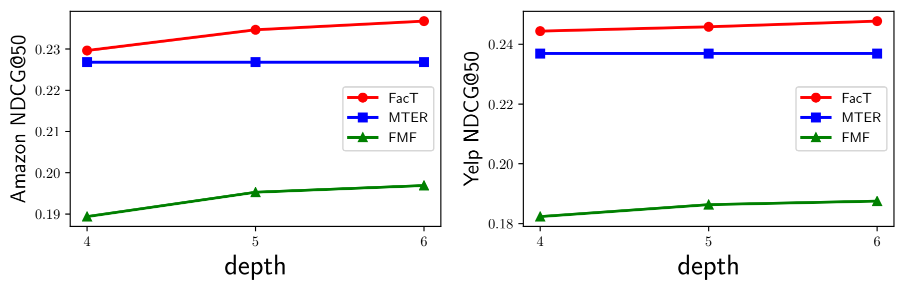


Figure 2.5: Varying the depth of user tree.

contribute more to the feature-based recommendation algorithm learning. In FacT, when the number of item features is limited, it cannot correctly create tree branches to guide latent factor learning. And more item features give FacT a higher degree of freedom to recognize the dependency between users and items.

Maximum tree depth

In FacT, we cluster the users and items along with the tree construction. The maximum tree depth controls the resolution of clusters, e.g., how many intermediate and leaf nodes will be created. We fixed all the other hyper-parameters and only tuned the maximum depth of each tree to verify the effect of it. The results are shown in Figure 2.5. We compared the performance of FacT with FMF and MTER. FMF introduces user tree construction to cluster users for cold-start recommendation. And MTER is the best baseline we had in Table 2.2, but as it does not have a tree structure, its performance remains constant in this experiment. And for FacT, we fixed the depth of item tree to 6 and varied the depth of user tree. We can observe both FMF and FacT got better performance with an increasing tree depth, which increases the granularity of the learned latent representations for users.

Cold-start Recommendation

Cold-start is an well-known and challenging problem in recommender systems. Without sufficient information about a new user, it is hard for a recommender system to understand the user's interest and provide recommendations with high quality. A by-product of FacT is that the rules learned in the user tree naturally serve as a set of interview questions to solicit user preferences when a new user comes to the system, i.e., cold-start. For example, based on the user tree in Figure 2.3, the system would get a good understanding of a new user by asking just a few questions following the paths on the tree. In this experiment, we study how FacT performs on the new users. First, we separated the users into two disjoint subsets, containing 95% and 5% users, for training and testing respectively. On the training set, we learned the model and built the user tree and item tree. During testing, for each testing user, we select their first k reviews to construct his/her item feature based user profile (i.e., F_i^u as defined in Eq Eq (2.2.3)). By matching against the user tree, we can easily find the leaf node for each testing user. Then, we use the latent factors reside in the selected leaf node to rank items for this user. We evaluate the performance in the remaining observations from the same user as ground-truth.

We compared FacT with FMF model as it is the only baseline that can handle cold-start. We varied the number of observations for each testing user from 0 to 5, and the results are shown in Figure 2.6. First, it is clear to observe that NDCG got improved with an increasing number of observations used to create the user profile for both FacT and FMF. This indicates the effectiveness of user clustering on the user tree in these two models. Second, thanks to the construction of item tree and BPR constraint, FacT got consistently better performance than FMF. In particular, NDCG@50 for FacT increases faster than FMF with more observations. We attribute this to the fact that FacT uses the item features and user opinions collected from the reviews to perform tree construction, while FMF only uses the item ratings to group users. This indicates the effectiveness of review information in modeling users.

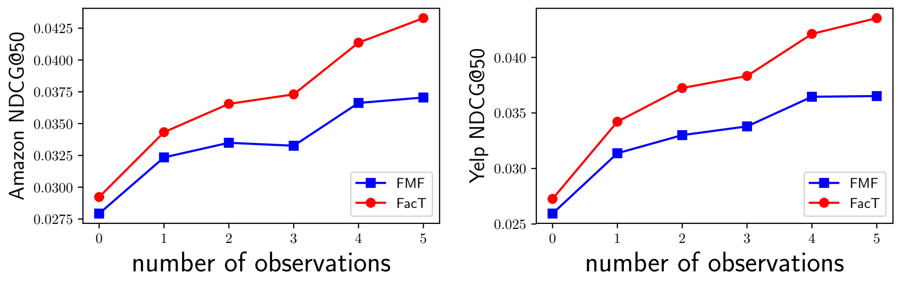


Figure 2.6: NDCG@50 v.s., the # observations in cold-start.

Table 2.4: Result of warm-start user study.

Average Score	Amazon			Yelp		
	EFM	MTER	FacT	EFM	MTER	FacT
Q1	3.64	3.96	**4.45***	3.45	4.06	**4.30***
Q2	3.48	3.88	**4.03**	3.40	3.87	**4.13**
Q3	3.07	3.02	**3.88***	2.98	3.26	**3.94***

* p-value < 0.05

2.2.4 User Study

We performed serious user studies to evaluate user satisfaction of both the recommendations and explanations generated by FacT. We evaluated the performance of FacT on both warm-start users, whose ratings and reviews are known to the system beforehand, and cold-start users who are totally new to the system. The study is based on the review data in Amazon and Yelp datasets used in previous experiments. We recruited participants on Amazon Mechanical Turk to interact with our system and collected their responses. To reduce the variance and noise of the study, we required the participants to come from an English-speaking country, older than 18 years, and have online shopping experience.

Warm-start Users

In the warm-start setting, we assume user's purchase history is known to the recommender system. However, we are not able to trace the participants' purchase history on Mechanical Turk. Instead, we performed a simulation-based study, in which we asked the participants to evaluate our system from the perspective of selected users in our datasets. Specifically, for each participant, we randomly selected a user from our review dataset and presented this user's reviews for the participant to read. The participants were expected to infer this user's preferences from the review content. Then the participant will be asked several questions to evaluate the recommendation and explanation generated by our algorithm.

We carefully designed the survey questions to evaluate different aspects of our recommender algorithm as follows:

Q1: Generally, are you satisfied with our recommendations?
Q2: Do the explanations presented to you really match your preference?
Q3: Do you have any idea about how we make recommendations for you?

We intended to use Q1 to evaluate user satisfaction of recommended items, use Q2 to judge the effectiveness of explanations, and use Q3 to evaluate the transparency of an explainable recommendation algorithm. For each question, the participants are required to choose from five rated answers: 1. Strongly negative; 2. Negative; 3. Neutral; 4. Positive; and 5. Strongly positive. We used EFM and MTER as baselines, since they both can provide textual explanations, and conducted A/B tests to ensure the evaluation is unbiased. Three hundred questionnaires were collected in total and the results are reported in Table 2.4.

Table 2.5: Results of cold-start interleaved test.

number of votes	Amazon		Yelp	
	FMF	FacT	FMF	FacT
Q1	44	**63***	40	**64***
Q2	43	**64***	34	**70***
Q3	45	**62**	33	**71***

* p-value < 0.05

From the statistics, FacT apparently outperformed both baselines in all aspects of this user study, which is further confirmed by the paired t-test. Comparing FacT with EFM and MTER on Q1, the improvement in offline validated recommendation quality directly translated into improved user satisfaction. For Q2, the advantage of FacT shows the effectiveness of our predicate selection in explanation rule construction, which captures user's underlying preferences. Moreover, the results on Q3 verified the user-perceived transparency of our tree guided recommendation and rule-based explanation mechanism.

Cold-start Users

Unlike warm-start users, cold-start users have no review history. In order to generate recommendation and explanation for these users, we progressively query user responses through an interview process. Specifically, each node of the user tree in FacT corresponds to an interview question: *"How do you like this [feature]?"*, where *[feature]* was learned to optimize the explanation rule at this node. When the user answers the interview question designated at the current node, he/she will be directed to one of its three child nodes according to the answer. As a result, each user follows a possibly different path from the root node to a leaf node during the interview process. A user's associated latent factor is adaptively refined at each intermediate node based on the user's responses. We make recommendations and explanations according to the resulting path. For comparison, FMF is set as a baseline, since it is the only algorithm that can address the cold-start problem with the same interview process as FacT. As FMF uses items instead of features to construct the tree, the interview question there is changed to *"How do you like this [item]?"*

To interview each participant in this user study, we developed a platform to let the participant interact with our system. [5] To increase the sensitivity of comparison between two recommendation algorithms, we conduct interleaved test [112] in this cold-start study. The participant was asked to interact with two models one after the other in a random order, to compare which one is better according to our designed questions. The recommendation is interactive, based on the participants' responses to the interview questions (i.e., traversing in the user tree). There are three questions for them to answer to compare the recommendations and explanations generated by these two algorithms:

Q1: Generally, between system A and B, whose recommendations are you more satisfied with?
Q2: Between system A and B, whose explanations do you think can better help you understand the recommendations?
Q3: Between system A and B, whose explanations can better help you make a more informed decision?

We collected more than 100 valid responses on each dataset and reported the results in Table 2.5. We can find that FacT is preferred than FMF in all questions on both datasets. It suggests that: First, feature-based rule construction is more effective than item-based rule construction, which leads to improved ranking quality in FacT. Second, the feature-based explanations are preferred than the item-based ones, as the former characterizes user preferences at a finer granularity. Last, feature-based explanation rules also provide improved transparency than item-based explanations, which verifies the explainability of our solution. All the evidences from this interleaved user study demonstrate the power of FacT to address the cold-start problem.

[5] https://aobo-y.github.io/explanation-recommendation/

Table 2.6: Case study on two explainable recommendation algorithms' output. Two restaurants are evaluated by the two algorithms, with corresponding recommendation scores and explanation output. We manually labeled attribute words in italic and sentiment words in bold.

Item	Algorithm 1 (Our proposed model)		Algorithm 2 (NRT [37])	
	Score	Explanation	Score	Explanation
A	4.2	the *sushi* is **good**, the *rolls* are **fresh** and the *service* is **excellent**.	4.1	their *prices* are **decent**, but the *portions* are **pretty small**.
B	2.1	it was a bit **loud** and the *service* was **slow**.	2.2	**great** *food*, **clean**, and **nice** *atmosphere*.

2.3 Sentiment Aligned Natural Language Explanation Generation

Due to the lack of explicit explanation training data, most explainable recommendation solutions appeal to user reviews as a proxy [12, 108, 110, 113–116]: a good explanation should overlap with user-provided reviews. This is backed by extensive prior research in sentiment analysis [117] that there is a strong correlation between opinion ratings and associated review content. But the approximation also inadvertently shifts the objective of explanation learning to generating or even memorizing reviews, in a verbatim manner. It unfortunately drives the current practice in explainable recommendation to decoupling the learning of recommendation and explanation into two loosely linked sub-problems with their own objectives (e.g., rating prediction vs., content reconstruction) [12, 37, 108]. But we have to emphasize that the content generated with fairly fluent language is not sufficient to be qualified explanations, as a good explanation must elaborate why the recommendation is particular to the user. Ideally, based on the provided explanations, a user should reach the same conclusion as the system does about why an item is recommended, i.e., explanation as a defense of the recommendation.

We believe the sentiment delivered by the explanation text needs to reveal the details of how items are scored and ranked differently by the system. We formulate this as *sentiment alignment* between the explanation text and system's corresponding recommendation. To demonstrate the importance of sentiment alignment, we compare example output from two explainable recommendation algorithms (one proposed in this work, and another from [37]) in Table 2.6. Both algorithms strongly recommended restaurant A over B, as suggested by the corresponding large margins in their recommendation scores. With Algorithm 1's explanations, one can easily recognize restaurant A is recommended because of better quality in its food and service. But on the contrary, it is much harder to comprehend the recommendations based on the Algorithm 2's explanations, as the presented difference become subtle, though their readability is comparable to Algorithm 1's. Two major reasons cost misaligned explanations in the second algorithm: 1) at training time, it only uses text reconstruction loss for explanation learning; 2) at inference time, the explanation is generated largely independently from the recommendation (as it only uses the predicted rating as an initial input for text generation). The failure to align sentiment conveyed in the explanation text with the recommendations not only cannot help users make informed decisions, but also makes them confused or even doubt about recommendations, which is totally against the purpose of explainable recommendation.

We propose to enforce sentiment alignment in *both* training and inference time for improved explainable recommendation. In particular, the learning of recommendation is modeled as a neural collaborative filtering problem [27], and the learning of explanation is modeled as a neural text generation problem [118]. We force the recommendation module to directly influence the learning of explanations by two means. First, we introduce two gated networks to our neural language model to fuse the intermediate output from the recommendation module to affect the word choice at every position of an explanation. Second, a stand-alone sentiment regressor is added in between the two modules' output, such that its predicted sentiment score on the explanation text should be close to the given recommendation score. When discrepancy occurs, the explanation module is pushed to minimize the difference. At inference time, all our treatments for sentiment alignment are kept. But since the explanation module has been learnt, the sentiment score gap is minimized by solving a constrained decoding problem.

We evaluate the proposed solution on both recommendation and explanation tasks, with particular focuses on the text quality, attribute personalization, and sentiment alignment of the generated explanations. Empirical results show that our solution improves the performance on both tasks, with particularly improved explanation quality via its enhanced sentiment alignment.

2.3.1 Related Work

User-provided reviews have been popularly used as a proxy of explanations in explainable recommendations [110, 113, 119]. One typical type of solutions directly extract representative text segments from existing reviews as explanations. For example, NARRE [110] uses attention to aggregate reviews to represent users and items for recommendation, in order to choose the most attentive reviews as explanations for each particular item. Wang et al. [113] extend the idea with reinforcement learning to extract the most relevant review text segments that match a given recommender system's rating prediction. However, such explanations are limited to existing reviews, some of which may not even be qualified as explanations (e.g., describing a personal experience). Moreover, these models only focus on selecting reviews to identify the items' characteristics, instead of addressing the reasons for a particular recommendation provided by the system.

Another family of solutions learn to generate explanations from reviews. Many of them learn to predict informative elements retrieved from reviews as explanations [108,114,120]. As a typical example, MTER [108] predicts items' feature words and corresponding users' opinion words alone with its recommendations. Its explanations are generated by placing the predicted words into predefined templates, which however lack necessary expressiveness and diversity of nature language. To address this deficiency, neural language models have been applied to synthesize natural language explanations [37, 115, 121, 122]. For example, NRT [37] models explanation generation and item recommendation with a shared user-item embedding space, where its predicted recommendation rating is used as part of the initial state for corresponding explanation generation. Neither the template-based or generation-based solutions paid enough attention to the sentiment alignment issue between recommendations and explanations. Although they jointly model recommendation and explanation (e.g., sharing embeddings), the objectives of training each module are still isolated. DualPC [116] realizes the importance of consistency between the two learning tasks, and introduces a duality regularization based on the joint probability of explanations and recommendations. However, the correlation imposed by duality does not have any explicit semantic meaning to the end users. In contrast, we require the output of models to be consistent in their carried sentiment, which is perceivable by an end user. Our solution treats explanation as a dependent of recommendation, and solves a constrained decoding problem to infer the most aligned explanation at testing time accordingly.

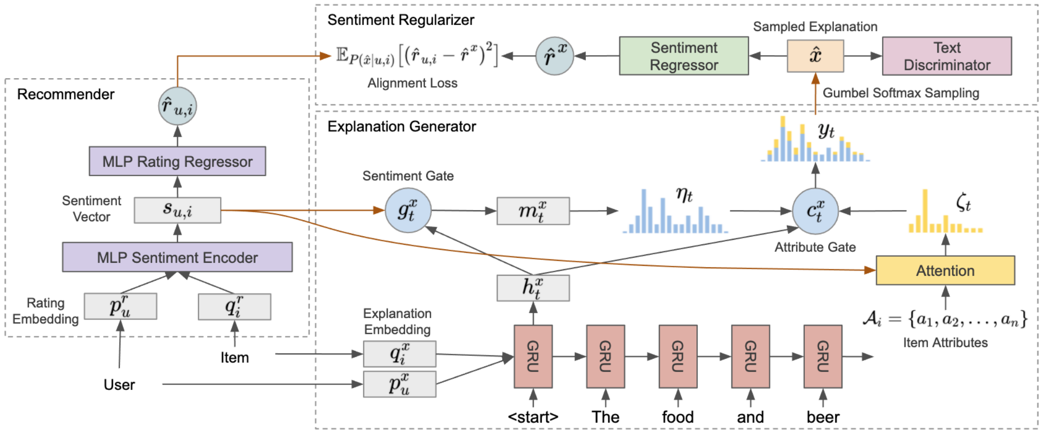


Figure 2.7: Model architecture of SAER. Sentiment alignment is enforced through three channels. First, SAER uses a shared sentiment vector to connect the recommender and explanation generator by the sentiment gate and attribute gate. Second, the sentiment regularizer samples generated explanations with Gumbel softmax and requires their carried sentiment (calculated by a pre-trained sentiment regressor) to match with the recommender's output score. Third, at inference time, constrained decoding is performed to ensure the alignment in the generated explanation. SAER also uses adversarial training to improve the explanations' readability in its sentiment regularizer.

2.3.2 Sentiment Aligned Explainable Recommendation

For a given pair of user u and item i, the model outputs a personalized recommendation based on its computed score $r_{u,i}$ and a word sequence $x_{u,i} = \{w_1, w_2, \ldots, w_n\}$ as its explanation. To learn such a model, we assume an existing training dataset, which includes a set of users $\mathcal{U}$, items $\mathcal{I}$, ratings $\mathcal{R}$, attributes $\mathcal{A}$, and explanation text $\mathcal{X}$, denoted as as $\{\mathcal{U}, \mathcal{I}, \mathcal{R}, \mathcal{A}, \mathcal{X}\}$. The attributes and explanations can be prepared from user-provided review corpora; and we will introduce the procedure we adopted for this purpose later in the experiment section.

We also define a vocabulary set $\mathcal{V} = \{w_1, w_2, ..., w_{|\mathcal{V}|}\}$ for explanation generation. We define attributes as items' popular properties mentioned in the review text, and thus they are a subset of vocabulary $\mathcal{A} \subset \mathcal{V}$.

Our model architecture for addressing explainable recommendation is shown in Figure 2.7. It consists of three major components: 1) recommender, which takes a user and item pair (u, i) as input to predict a recommendation score $\widehat{r}_{u,i}$, which measures the affinity between u and i; 2) explanation generator, which takes the (u, i) pair as input and generates a word sequence $\widehat{x}_{u,i} = \{w_1, w_2, \ldots, w_n\}$ as the corresponding explanation; and 3) sentiment regularizer, which measures sentiment alignment between the generated explanation and recommendation. All three components closely interact with each other at *both* training and inference time for improved explanation generation, especially for enhanced sentiment alignment. We name our solution Sentiment Aligned Explainable Recommendation, or SAER in short. Next, we will zoom into each component to introduce its design principle and technical details.

Personalized Recommendation

As our focus in this work is not to design yet another recommendation algorithm, we adopted the neural collaborative filtering solution for the purpose [27]. Arguably any latent factor models that explicitly learn user and item representations [12, 99, 108] can be adopted. We stack two Multi-Layer Perceptron (MLP) networks to predict the recommendation score $\widehat{r}_{u,i}$ for a given (u, i) pair. The first MLP encodes the (u, i) pair to a latent sentiment vector $\boldsymbol{s}_{u,i} \in \mathbb{R}^{d_s^r}$, and the second MLP maps the sentiment vector $\boldsymbol{s}_{u,i}$ into the numerical rating $\widehat{r}_{u,i}$. We refer to the first MLP as *sentiment encoder* and the second one as *rating regressor*. Instead of using the predicted score $\widehat{r}_{u,i}$ to influence explanation generation, we choose to inform the explanation generator by the encoded sentiment vector $\boldsymbol{s}_{u,i}$. We defer the details of this design to the next section. In the recommendation module, we define the latent embedding matrices for users and items as $P^r \in \mathbb{R}^{d^r \times |\mathcal{U}|}$ and $Q^r \in \mathbb{R}^{d^r \times |\mathcal{I}|}$ respectively, where d^r is the dimension of the embedding vectors. The sentiment encoder concatenates the embedding vector p_u^r and q_i^r as its input and passes it through multiple layers with leaky ReLU activation to get the sentiment vector $\boldsymbol{s}_{u,i}$ encoded. Besides its use in the explanation generator, $\boldsymbol{s}_{u,i}$ is then mapped by the rating regressor through another set of multi-layer leaky ReLUs to get the final recommendation score $\widehat{r}_{u,i}$.

In addition to the popularly used Minimal Squared Error (MSE) [37, 110] to train our recommender, we also introduce a pairwise hinge loss to improve the trained recommender's ranking performance. Specifically, for each user u, we collect a set of personalized item pairs $\mathcal{B}_u = \{(i, j)|r_{u,i} > r_{u,j}\}$, where i and j are two items rated by user u and one is preferred than another as observed in the training dataset. We did not use the popular BPR loss [98], because it tends to push ratings to extreme values, which is inconsistent with our sentiment regularizer's requirement to be explained later.

Based on the rating set $\mathcal{R}$ and personalized item pair set $\{\mathcal{B}_u\}_{u \in \mathcal{U}}$, the loss for recommender training is defined as:

$$L^r = \frac{1}{|\mathcal{R}|} \sum_{r_{u,i} \in \mathcal{R}} \left(\widehat{r}_{u,i} - r_{u,i}\right)^2 + \sum_{u \in \mathcal{U}} \frac{\lambda_h}{|\mathcal{B}_u|} \sum_{(i,j) \in \mathcal{B}_u} \max\left(0, \beta - (\widehat{r}_{u,i} - \widehat{r}_{u,j})\right)$$

where $\beta > 0$ is a hyper-parameter to control the separation margin, i.e., it penalizes the model when the predicted difference between $\widehat{r}_{u,i}$ and $\widehat{r}_{u,j}$ is smaller than β, and λ_h is the coefficient to control the balance between MSE loss and pairwise hinge loss.

Explanation Generation

Motivated by the success of neural language generation, we appeal to a Recurrent Neural Network (RNN) model with Gated Recurrent Units (GRUs) [123] for explanation generation. To make the generation related to the user and item, we first map the input user u and item i to their embeddings p_u^x and q_i^x with the latent matrices $P^x \in \mathbb{R}^{d^x \times |\mathcal{U}|}$ and $Q^x \in \mathbb{R}^{d^x \times |\mathcal{I}|}$ learnt by the explanation generator. We should note this set of embeddings are different from those used in the recommender (i.e., P^r and Q^r), as they should characterize different semantic aspects of users and items (ratings vs., text). We hence use superscript x to indicate variables and parameters related to explanation generator. To generate explanation text, the embeddings are concatenated and linearly converted into the initial RNN hidden state; and then the GRU generates hidden state $\boldsymbol{h}_t^x \in \mathbb{R}^{d_h^x}$ at position t with previous state $\boldsymbol{h}_{t-1}^x$ and input word w_t, and predicts the next word w_{t+1} recursively.

Though similar model design has been used for explanation generation [37, 116], this straightforward application of RNN can hardly generate satisfactory explanations, where two issues are left open. First, a good explanation is expected to be personalized and specific about the recommendation; generic content, such as "*this is a nice restaurant*" can never be informative. It is important to explain the recommended item by the user's most concerned attributes. Second, the sentiment carried in the explanation, especially on the mentioned attributes, should be explicit and consistent with the recommendation (as shown in our case study in Table 2.6). There is no guarantee that a simple RNN can satisfy both requirements.

We enhance our generator design with two gated sub-networks upon GRU to address the aforementioned issues. First, we design a sub-network, named attribute gate, to guide attribute word generation with respect to the input user-item pair and the predicted recommendation sentiment. The attribute gate is built based on a pointer network (or copy mechanism) [124, 125], which decides whether the current position should mention an attribute word and the corresponding distribution of attribute words based on the generation context. To make the choice of attribute word specific to the item, for each item i we build an attribute set with all attribute words that appear in i's associated training explanation text: $\mathcal{A}_i = \{a_k | a_k \in \{x_{u,i} | u \in \mathcal{U}\}\}$. To make the attribute choice depend on the already generated content, we attend on the concatenation of the current position's RNN hidden state $\boldsymbol{h}_t^x$ and sentiment vector $\boldsymbol{s}_{u,i}$ to compute the distribution of these attribute words,

$$\mathbf{z}_{t,k} = [\boldsymbol{h}_t^x, \boldsymbol{s}_{u,i}]^\top W_z^x \mathbf{v}_{a_k}, \forall k, a_k \in \mathcal{A}_i; \;\; \zeta_t = softmax(z_t), \tag{2.3.1}$$

where $W_z^x \in \mathbb{R}^{(d_h^x + d_s^r) \times d_\mathbf{v}^x}$ and $\mathbf{v}_{a_k}$ is the word embedding of attribute a_k. $\mathbf{z}_{t,k}$ is computed for every a_k in $\mathcal{A}_i$, i.e., $\mathbf{z}_t = \{z_{t,1}, z_{t,2}$
$...z_{t,|\mathcal{A}_i|}\}$. ζ_t is the resulting attribute word distribution at position t. For better performance, an extra linear transformation can be applied to $\boldsymbol{h}_t^x$ to compress it into a lower dimension before computing attention, which helps avoid overfitting attentions to the text generation context but ignoring the sentiment context.

To decide if we need to generate an attribute word using Eq Eq (2.3.1) at position t, we compute the copy probability with respect to the current context $\boldsymbol{h}_t^x$ by $c_t^x = \sigma(W_c^x \boldsymbol{h}_t^x + b_c^x)$, where $\sigma(\cdot)$ is the sigmoid function, $W_c^x \in \mathbb{R}^{d_h^x}$ and $b_c^x \in \mathbb{R}$. c_t^x allows us to mix the vocabulary distribution predicted by GRU and attribute word choice to get our final word distribution at position t.

Second, we design a sentiment gate to fuse the sentiment vector $\boldsymbol{s}_{u,i}$ to align sentiment in the generated explanation text. Our key insight is that not all words convey sentiment, we need to choose the right word at the right place to express consistent sentiment as needed by the recommender. Similar to our attribute gate design, we apply a soft gate to decide how each position is related to the intended sentiment. At position t, the sentiment gate calculates a ratio g_t^x with respect to the RNN's hidden state $\boldsymbol{h}_t^x$. The sentiment vector $\boldsymbol{s}_{u,i}$ is then weighted and merged with $\boldsymbol{h}_t^x$,

$$g_t^x = \sigma(W_g^x \boldsymbol{h}_t^x + b_g^x), \;\; \mathbf{m}_t^x = tanh\big(\boldsymbol{h}_t^x + g_t^x(W_m^x \boldsymbol{s}_{u,i} + b_m^x)\big) \tag{2.3.2}$$

where $W_g^x \in \mathbb{R}^{d_h^x}$ and $b_g^x \in \mathbb{R}$ produce a scalar g_t^x. $\mathbf{m}_t^x$ is the sentiment fused latent vector to predict the vocabulary distribution for position t. Because not all words are about sentiment, to better differentiate the positions where the intended sentiment needs to be expressed from the rest, we impose sparsity on the learned gate value g_t^x using L1 regularization at training time. In other words, the gate is open only when necessary.

We compute the final word distribution by consolidating the outputs of the two gated sub-networks (Eq Eq (2.3.1) and Eq (2.3.2)). First, the sentiment fused latent vector $\mathbf{m}_t^x$ is fed through a linear layer to calculate the vocabulary distribution $\eta_t = softmax(W_\mathbf{v}^x \mathbf{m}_t^x + b_\mathbf{v}^x)$, where $W_\mathbf{v}^x \in \mathbb{R}^{|\mathcal{V}| \times d_h^x}$ and $b_\mathbf{v}^x \in \mathbb{R}^{|\mathcal{V}|}$. Second, the vocabulary distribution η_t and attribute word distribution ζ_t are merged to obtain the final word distribution with respect to the copy probability c_t^x, i.e., $\mathbf{y}_t = (1 - c_t^x)\eta_t + c_t^x \zeta_t$, where the value of w_k in ζ_t is 0 if w_k is not an attribute word.

The objective for explanation generation is to minimize the negative log-likelihood loss (NLL) on the training explanation set $\mathcal{X}$,

$$L^x = -\sum_{x \in \mathcal{X}} \sum_{w_t \in x} \log \mathbf{y}_t(w_t) + \lambda_g \sum_{x \in X} \sum_{w_t \in x} |g_t^x|$$

where $\mathbf{y}_t(w_t)$ is the resulting probability of word w_t and λ_g is the coefficient for the L1 regularization of the sentiment gate values.

Sentiment Alignment

Though our sentiment gate design (Eq Eq (2.3.2)) introduces predicted sentiment from the recommender to the explanation generator, it is still insufficient to guarantee sentiment alignment, for three major reasons. First, word-based NLL training cannot maintain the whole sentence's sentiment. This weakens its prediction quality on sentiment words. Second, the explanation generator might utilize the sentiment vector differently as the recommender does, so that the recommendation rating might diverge from the sentiment carried by the explanation. Third, the generation process at the inference stage works differently from the training stage [126]: at inference time, the previously decoded word is used as the input for the next word prediction, instead of the ground-truth word as at the training time. Hence, the learnt text pattern might not be fully exploited at the inference time.

We introduce the sentiment regularizer to close the loop between the recommender and explanation generator. It uses a stand-alone sentiment regressor to predict the sentiment rating $\widehat{r}^x$ on the generated explanation text $\widehat{x}_{u,i}$ for user-item pair (u, i), and requires the explanation generator to match the rating $\widehat{r}_{u,i}$ from the recommender accordingly. We do not have any particular assumption about the sentiment regressor; and any state-of-the-art regression models can be leveraged [117]. In this work, we employed an MLP on top of a bidirectional RNN text encoder with inner attention for rating regression, and denote it as $f^R(x) \to r^x$. We pre-train this regressor based on ground-truth $\{\mathcal{R}, \mathcal{X}\}$ in the training set; and fix the learnt model thereafter.

To enforce sentiment alignment by the predicted ratings, we introduce a new loss to the training of our explanation generator,

$$L^a = \sum_{u \in \mathcal{U}, i \in \mathcal{I}} \mathbb{E}_{P(\widehat{x}|u,i)} \big[(\widehat{r}_{u,i} - f^R(\widehat{x}))^2 \big] \tag{2.3.3}$$

where $P(\widehat{x}|u, i)$ is the probability of generating $\widehat{x}$ for the given u and i. We should note this loss is not necessarily restricted to the observed (u, i) pairs in the training set; instead, it could be any pairs of them, since both the recommender and explanation generator can generate output on any given (u, i) pair. It thus enables data augmentation for sentiment alignment.

However, because the word distribution is categorical, the generation of $\widehat{x}$ is not differentiable. It makes direct optimization with respect to Eq Eq (2.3.3) infeasible. As a result, we appeal to Gumbel softmax [127] to obtain approximated gradient of sampling from a categorical distribution. Briefly, Gumbel softmax reparameterizes the randomness in sampling by a gumbel distribution and simulates a relaxed one-hot vector with softmax. As we need a strict one-hot vector to represent each single word, we adopt the Straight-Through (ST) Gumbel softmax estimator [127]. For each (u, i) pair in Eq Eq (2.3.3), we back-propagate the gradient from L^a to the explanation generator to improve the quality of sentiment alignment on the whole sequence.

Unfortunately, this new sentiment alignment loss might also attract the generation process to produce unreadable sequences, which however match the intended sentiment ratings. For example, the sentiment regressor may give a very positive rating to an unnatural sentence "*good good good good*", when the recommender also happens to predict a high rating for this item. To improve the readability of our generated explanation, we introduce a text discriminator f^D, which learns to differentiate the authentic explanations from the generated ones, to guide the explanation generation as well. Our design allows any text classifier. In this work, we used an MLP binary classifier on top of a bidirectional RNN encoder for the purpose. We train the discriminator using cross-entropy loss with the ground-truth explanations x as positive and the generated explanations $\widehat{x}$ as negative,

$$L^D = -\frac{1}{|\mathcal{X}|} \sum_{x \in \mathcal{X}} \log f^D(x) - \sum_{u \in \mathcal{U}, i \in \mathcal{I}} \mathbb{E}_{P(\widehat{x}|u,i)} \big[\log(1 - f^D(\widehat{x})) \big]$$

Correspondingly, another objective of explanation generation is to fool the discriminator, i.e., the adversarial loss,

$$L^c = -\sum_{u \in \mathcal{U}, i \in \mathcal{I}} \mathbb{E}_{P(\widehat{x}|u,i)} \big[\log f^D(\widehat{x}) \big]$$

This loss also requires sampled explanations $\widehat{x}$ as the input, like the alignment loss defined in Eq Eq (2.3.3). The same Gumbel softmax sampling technique is used for end-to-end training.

As we pointed out before, addressing the sentiment alignment issue in training alone is still insufficient, we introduce a constraint-driven decoding strategy to enhance the alignment at the inference stage as well. Similarly as in training, we use MSE to quantify the difference between the rating predicted from the explanation text and that from the recommender. Because the sentiment regressor can only be applied to a complete sequence, the search space is too large to enumerate by the generator. Hence, we treat generating explanation $\widehat{x}$ at inference time as a sequence of decision making, where each action is to generate a word w_t at position t, given its already generated prefix as state. But we do not have feedback on the actions, until we complete $\widehat{x}$; and the return for taking the series of actions can be measured by $Q(\widehat{x};\widehat{r}_{u,i}) = [\widehat{r}_{u,i} - f^R(\widehat{x})]^2$. To find a policy that minimizes return, we need to estimate the value function under each state. This can be effectively addressed by Monte Carlo Tree Search (MCTS) [128]. Basically, we estimate the value function using our trained explanation generator for roll-out. When at position t for generating $\widehat{x}_{u,i}$, we will sample n complete sequences for every action w using the current prefix $\{w_1, w_2, \dots, w_{t-1}\}$, following the distribution specified by the explanation generator: $\widehat{X}_{u,i,t}(w) = \big\{\widehat{x}_k = MCTS_{u,i}(w_1, w_2, \dots, w_{t-1}, w)\big\}_{k=1}^{n}$. Then the value of taking action w at position t can be estimated by,

$$Q(w_1, w_2, \dots, w_{t-1}, w; \widehat{r}_{u,i}) = \frac{1}{|\widehat{X}_{u,i,t}(w)|} \sum_{\widehat{x}_k \in \widehat{X}_{u,i,t}(w)} Q(\widehat{x}_k, \widehat{r}_{u,i})$$

Based on the estimated values, we can take the action that minimizes the value. We integrate our MCTS with top-k sampling, i.e., at each decoding position t, we sample k most likely words according to word distribution $\mathbf{y_t}$ and then use MCTS to select the one that minimizes the estimated value under given state.

A vanilla implementation of MCTS is expected to be expensive and slow in our problem, as it needs to complete the sequence at each position from an RNN model for multiple times. Fortunately, our sentiment gate design provides a short path for efficient sampling: as sentiment is only carried by a small number of words, there is no need to conduct such expensive sampling procedure at every position. Instead, we only need to perform MCTS at positions where sentiment is expressed. Hence, we set a threshold on the sentiment gate's value to decide when to perform MCTS. When the gate's value is below the threshold, we will directly sample from the top-k words of the explanation generator's prediction.

End-to-End Model Training

Putting together the three components in our proposed explainable recommendation solution SAER, the overall objective of our model training is formulated as:

$$J = \min_{\Theta} \left(\lambda_r L^r + \lambda_x L^x + \lambda_a L^a + \lambda_c L^c + \lambda_n ||\Theta||^2\right)$$

where Θ is the complete set of model parameters, and $\{\lambda_r, \lambda_x, \lambda_a, \lambda_c\}$ are the corresponding coefficients to control the relative importance of each component in model training. We also include an L2 regularization for the model parameters Θ, weighted by its coefficient λ_n. The parameters are then effectively estimated end-to-end with stochastic gradient optimizer of Adam [129].

We split the whole training process into five stages. First, estimate the sentiment regressor on $\{\mathcal{X}, \mathcal{R}\}$, as it does not depend on the other parts of our model. Second, pre-train the recommender on $\{\mathcal{U}, \mathcal{I}, \mathcal{R}\}$ till convergent. This step is essential to learn a good sentiment encoder whose output will be used to inform the explanation generator. Third, freeze the recommender and train the generator on $\{\mathcal{U}, \mathcal{I}, \mathcal{A}, \mathcal{X}\}$ with negative log-likelihood loss only. Fourth, after the separate training converges, start joint training of the recommender and explanation generator. This step allows the model to align the sentiment representation from both modules. At last, freeze the recommender, and turn on the sentiment regularizer to further improve the explanation generator. At this stage, the explanation discriminator and generator are trained in turn.

Table 2.7: Statistics of the processed datasets.

Dataset	# Users	# Items	# Reviews	# Attributes
Yelp	15,642	21,525	1,108,971	498
Ratebeer	3,895	6,993	1,073,762	333

Table 2.8: Evaluation of personalized recommendation in terms of rating prediction (RMSE, MAE) and item ranking (NDCG).

	Yelp				Ratebeer			
Model	RMSE	MAE	NDCG@5	NDCG@10	RMSE	MAE	NDCG@5	NDCG@10
NMF	1.1034	0.8164	0.5067	0.7344	2.2228	1.6609	0.6334	0.7766
SVD	1.0286	0.7975	0.5246	0.7519	2.2942	1.6474	0.6120	0.7593
NCF	1.0532	0.8251	0.5150	0.7420	2.0857	1.5002	0.6621	0.8004
NARRE	1.0275	0.8035	0.5230	0.7509	2.0714	1.4975	0.6641	0.8030
NRT	1.0254	0.8017	0.5262	0.7540	2.0743	1.4922	0.6620	0.8008
SAER	**1.0190**	**0.7948**	**0.5278**	**0.7553**	**2.0628**	**1.4842**	**0.6648**	**0.8034**

2.3.3 Experimental Evaluation

We quantitatively evaluate our model's performance on personalized recommendation and explanation generation in two different domains: restaurant recommendation on Yelp reviews [6] and beer recommendation on Ratebeer reviews [130]. Our model is compared against a set of state-of-the-art baselines on both offline data and user studies, where encouraging improvements are obtained.

Experiment Setup

Data Pre-Processing. We use the Sentires toolkit [131] to extract attribute words from reviews and manually filter out inappropriate ones based on domain knowledge. Although reviews are directly treated as explanations in many previous studies [110, 113], a recent work [122] suggests a large portion of review content is only about subjective emotion and thus does not qualify as explanations, e.g., "*I love the food*". An informative explanation should depict the details of items, e.g., their attributes, to help users perceive the exact reason behind recommendations, e.g., "*the fish is fresh*". Therefore, we restrict ourselves to sentences containing attribute words as explanations in our experiments. On top of the crafted explanations, we select 20,000 most frequent words and map others to unknown to build the vocabulary. Finally, as lots of users and items only have very few reviews in the datasets, we apply recursive filtering as in [108] to refine the datasets and alleviate this sparsity issue. The resulting statistics of the datasets are summarized in Table 2.7.

Baselines. To evaluate the personalized recommendation performance, we used the following baselines:

- **NMF**: Non-negative Matrix Factorization [132]. A widely used latent factor model, which decomposes the rating matrix into lower dimensional matrices with non-negative factors.
- **SVD**: Singular Value Decomposition [133]. It utilizes rating matrix as input for learning user and item representations.
- **NCF**: Neural Collaborative Filtering [27]. It is a modified matrix factorization solution which adopts neural networks to model the nonlinear vector operations.

We also include two explainable recommendation baselines that can output natural language sentences as explanations for comparing both the recommendation and explanation quality:

- **NARRE**: Neural Attentional Regression model with Review-level Explanations [110]. It learns the usefulness of the existing reviews through attention, and incorporates the review to enrich user and item representations for rating prediction. To fit in our evaluation, we select sentences from its most attentive reviews as explanations.
- **NRT**: Neural Rating and Tips Generation [37]. A multi-task learning solution for rating regression and content generation. It uses the predicted recommendation score to create initial states for content generation.

[6]https://www.yelp.com/dataset

Table 2.9: BLEU scores of generated explanations.

Dataset	Model	BLEU-1	BLEU-2	BLEU-4
Yelp	NARRE	20.46	5.72	2.12
	NRT	26.25	8.84	2.97
	SAER (topk)	27.43	9.53	3.18
	SAER (reg + topk)	28.69	10.29	3.37
	SAER	**28.88**	**10.44**	**3.44**
Ratebeer	NARRE	29.78	9.47	3.27
	NRT	42.16	17.54	5.63
	SAER (topk)	43.92	19.60	6.56
	SAER (reg + topk)	45.69	21.09	7.02
	SAER	**46.01**	**21.60**	**7.32**

Table 2.10: Performance of attribute prediction in generated explanations.

	Yelp		Ratebeer	
Model	Precision	Recall	Precision	Recall
NARRE	0.1415	0.1906	0.2176	0.2245
NRT	0.1791	0.1997	0.3443	0.1720
SAER (topk)	0.2024	0.2297	0.3523	0.2554
SAER (reg + topk)	0.1992	0.2319	0.3549	0.2614
SAER	**0.2115**	**0.2391**	**0.3702**	**0.2677**

Quality of Personalized Recommendations

We evaluate the recommendation quality both in terms of rating prediction (by RMSE and MAE) and item ranking performance (by NDCG@{5,10} [134]). The results are shown in Table 2.8. SAER demonstrates better performance in all metrics on both datasets. The performance difference among NCF, NRT and SAER is worth noting. Although their rating prediction modules all use MLP, NRT and SAER additionally leverage the content information for improved recommendation quality. Improvements from SAER against NARRE and NRT demonstrate that our sentiment vector and corresponding soft gate design better distill and exploit review data for joint learning.

Quality of Generated Explanations

We evaluate the quality of our generated explanations from three perspectives: text quality, attribute personalization, and sentiment alignment. We introduce two variants of our model to better analyze the effects of our sentiment regularizer and constrained decoding strategy. (1) SAER (topk), it removes sentiment regularization and decodes by top-k sampling, such that sentiment alignment is only introduced by the soft gates, without the alignment loss, nor the constrained decoding; (2) SAER (reg + topk), it uses sentiment regularization (i.e., the alignment loss) and decodes by top-k sampling, such that sentiment alignment is only enforced at training time.

Quality of Generated Text. We measure the quality of generated explanation text with BLEU [135], and report the results in Table 2.9. The extraction-based NARRE performed clearly worse than other generation-based models. This is because the synthesized natural language explanations are not limited to the existing review content and is more flexible to customize for a particular user-item pair. NRT uses the predicted ratings in the initial state for content generation, in comparison to the sentiment vectors used in SAER. The performance gap between NRT and SAER (topk) suggests that our sentiment vectors are more expressive and the two soft gates can better guide explanation generation throughout the process, than only affecting RNN's initial state. The additional gain brought by the sentiment regularizer in SAER (reg + topk) and constrained decoding in SAER highlights the benefits of sentiment alignment in both training and inference time.

Attribute Personalization. An informative explanation should cover the users' most concerned aspects. We evaluate such performance in terms of attribute personalization. For each user-item pair, we evaluate precision and recall of attribute word in the algorithms' explanations against ground-truth explanations. The results in Table 2.10 show the improvement brought by our attribute gate, which is proved to be effective in predicting users' most concerned attributes.

Table 2.11: Sentiment alignment evaluation of decoded explanations by RMSE. PD is the RMSE between explanation rating and predicted rating, and GT is the RMSE between explanation rating and ground-truth rating.

	Yelp		Ratebeer	
	PD	GT	PD	GT
NARRE	1.0932	1.4950	2.0996	2.9641
NRT	0.6676	1.2086	2.3302	3.1304
SAER (topk)	0.6908	1.2216	2.1727	3.0026
SAER (reg + topk)	0.6242	1.1849	1.6985	2.6769
SAER	**0.5505**	**1.1503**	**1.5911**	**2.6042**

Sentiment Alignment Between Ratings and Explanations. Offline evaluation of sentiment alignment is not easy, since it should be evaluated by the end users who receive the recommendation and explanation. In addition to depending on user studies to evaluate this aspect (reported in the next section), we also use our pre-trained sentiment regressor for an approximated offline evaluation. For a generated explanation, we infer its carried sentiment by our sentiment regressor. We then compute the RMSE between the inferred rating from explanation and that predicted by the recommendation module (marked as PD). This measures sentiment difference between the recommendation and corresponding explanation. We also compare the inferred rating against the ground-truth rating (marked as GT) as a reference. The results are presented in Table 2.11. Without our sentiment regularizer, SAER (topk) can already significantly outperform the baselines on Yelp, which demonstrates the utility of our two gated network design for sentiment alignment. And the alignment loss and constrained decoding further push SAER's explanations closer to its recommendations. Compared to the ground-truth rating, sentiment carried by the explanation is closer to the recommender's prediction. We hypothesize that this can be caused by the difficulty to predict ground-truth rating: as reported in Table 2.8, the accuracy of the recommender's rating prediction is at around the same level.

2.4 Comparative Explanation Generation for Recommendation

When being presented with a list of recommendations, typically sorted in a descending order, a user needs to make a choice. In other words, the provided explanations should help users *compare* the recommended items. Existing explainable recommendation solutions are not optimized to help users make such comparative decisions for two major reasons. First, the explanation of a recommended item is often independently generated without considering other items in the recommendation list. Second, the popularly adopted neural text generation techniques are known to be flawed of its generic content output [136, 137]. Particularly, techniques like maximum likelihood training and sequence greedy decoding lead to short and repetitive sentences composed of globally frequent words [138]. Such generic content cannot fulfill the need to differentiate the recommended items.

In this work, we tackle the problem of comparative explanation generation to help users understand the comparisons between the recommended items. We focus on explaining how one item is compared with another; then by using a commonly shared set of items as references (e.g., items the user has reviewed before), the comparisons among the recommended items emerge. Our solution is designed to generically work on top of other existing recommender systems. We do not have any assumptions about how the recommendation algorithm ranks items (e.g., collaborative filtering [139] or content-based [140]), but only require it to provide a ranking score for each item to our model (i.e., ordinal ranking) which reflects a user's preference over the recommended item.

To be specific, we design an extract-and-refine text generation architecture [138, 141] to explain the ranked items one at a time to the user, conditioned on their recommendation scores and associated reviews. We refer to the item to be explained in the ranked list as the target item, and user we are explaining to as the target user. First, the model extracts one sentence from the existing review sentences about the target item as a prototype, with a goal to maximize the likelihood of fitting the comparisons against the reviews written by the target user for other reference items. Then we refine the extracted prototype through a generative model to further polish the content for the target user. In this two stage procedure, the extraction module exploits the content already provided about the target item to ensure the relevance of generated explanations (e.g., avoid mentioning features that do not exist in the target item); and the refinement module further improve the explanation (e.g., informativeness and diversity of content) beyond the limitation of the existing content. We design a new explanation quality metric based on BLEU to guide the end-to-end training of the two modules, with a particular focus to penalize short and generic content in generated explanations. We compared the proposed solution with a rich set of state-of-the-art baselines for explanation generation on two large-scale recommendation datasets. Besides, we also conducted extensive user studies to have the generated explanations evaluated by real users. Positive results obtained on both offline and online experiments suggested the effectiveness of comparative explanations in assisting users to better understand the recommendations and make more informed choices.

2.4.1 Related Work

This work is closely related to two studies, DualPC [116] and SAER [38], which focus on strengthening the relation between recommendations and explanations. Specifically, DualPC introduces duality regularization based on the joint probability of explanations and recommendations to improve the correlation between recommendations and generated explanations. SAER introduces the idea of sentiment alignment in explanation generation. However, both of them operate in a *pointwise* fashion, i.e., independent explanation generation across items. Our solution focuses on explaining the comparisons between items. We should also emphasize our solution is to explain the comparison among a set of recommended items, rather than to find comparable items [142, 143].

2.4.2 Comparative Explanation Generation

Item recommendation in essence is a ranking problem: estimate a recommendation score for each item under a given user and rank the items accordingly, such that the utility of the recommendations can be maximized [144,145]. Instead of explaining how the recommendation scores are obtained, our work emphasizes on explaining how the comparisons between the ranked items are derived.

To learn the explanation model, we assume an existing corpus of item reviews from the intended application domain (e.g., hotel reviews). Each review is uniquely associated with a user u and an item c, and a user-provided rating r_c^u suggesting his/her opinion towards the item. We group the reviews associated with user u to construct his/her profile $\Omega_u = \{(x_1^u, r_1^u), (x_2^u, r_2^u), ..., (x_m^u, r_m^u)\}$, where x_i^u is the i-th review sentence extracted from user u's reviews and r_i^u is the corresponding opinion rating. r_i^u can be easily obtained when the detailed aspect ratings are available; otherwise off-the-shelf sentiment analysis methods can be used for the purpose (interested users can refer to [12, 108] for more details). As regards cold-start for users without reviews, generic profiles can be used instead which sample reviews from similar users clustered by other non-review-related features, such as rating history. We create the item profile as $\Psi_c = \{x_1^c, x_2^c, ..., x_n^c\}$, where x_j^c is the j-th review sentence extracted from item c's existing reviews. Unlike the user profile, the item profile does not include ratings. This is because the ratings from different users are not directly comparable, as individuals understand or use the numerical ratings differently. Our solution is agnostic to the number of entries in user profile Ω_u and item profile Ψ_c in each user and item.

We impose a generative process for a tuple (x, r_c^u) from user u about item c conditioned on Ψ_c and Ω_u. We assume when user u is reviewing item c, he/she will first select an existing sentence from Ψ_c that is mostly related to the aspect he/she wants to cover about the item. Intuitively, this can be understood as the user will first browse existing reviews of the item to understand how the other users evaluated this item. Then he/she will rewrite this selected sentence to reflect his/her intended opinion and own writing style. This can be considered as *a set to sequence generation problem*. For our purpose of explanation generation, we only concern the generation of opinionated text x. Hence, we take opinion rating r_c^u as input, which leads us to the following formulation,

$$P(x|u, c, r_c^u) = \sum_{x_j^c \in \Psi_c} P_{ref}(x|x_j^c, r_c^u, \Omega_u) P_{ext}(x_j^c|r_c^u, \Omega_u) \tag{2.4.1}$$

where $P_{ext}(x_j^c|r_c^u, \Omega_u)$ specifies the probability that x_j^c from item profile Ψ_c will be selected by user u, and $P_{ref}(x|x_j^c, r_c^u, \Omega_u)$ specifies the probability that user u will rewrite x_j^c into x. We name the resulting model Comparative Explainer, or CompExp in short.

In Eq Eq (2.4.1), $P_{ext}(x_j^c|r_c^u, \Omega_u)$ is essential to capture the comparative textual patterns embedded in user u's historical opinionated text content. To understand this, we can simply rewrite its condition part: define $\Delta r_i^u = r_c^u - r_i^u$, we have $(r_c^u, \Omega_u) = \{(x_i^u, \Delta r_i^u)\}_{i=1}^m$; hence, $P_{ext}(x_j^c|r_c^u, \Omega_u)$ characterizes whether the sentence x_j^c about item c is qualified to characterize the desired opinion difference conditioned on user u's historical content Ω_u and target rating r_c^u. For example, a negative Δr_i^u suggests the opinion conveyed in x_j^c is expected to be less positive than that in x_i^u. On a similar note, $P_{ref}(x|x_j^c, r_c^u, \Omega_u)$ quantifies if x is a good rewriting of x_j^c to satisfy the desired opinion rating r_c^u for item c by user u.

One can parameterize $P_{ext}(x_j^c|r_c^u, \Omega_u)$ and $P_{ref}(x|x_j^c, r_c^u, \Omega_u)$ and estimate the corresponding parameters based on the maximum likelihood principle over observations in Ω_u. However, data likelihood alone is insufficient to generate high-quality explanations, as we should also emphasize on fluency, brevity, and diversity of the generated explanations. To realize this generalized objective, assume a metric $\pi(x|u, c)$ that measures the quality of generated explanation x for user u about item c, the training objective of CompExp is set to maximize the expected quality of its generated explanations under $\pi(x|u, c)$,

$$J = \mathbb{E}_{x \sim P(x|u,c,r_c^u)}[\pi(x|u, c)] \tag{2.4.2}$$

$$\widehat{x}_j^c$$

In this work, we present a customized BLUE score specifically for the comparative explanation generation problem to penalize short and generic content.

Next, we dive into the detailed design of CompExp in Section 2.4.2, then present our metric $\pi(x|u, c)$ for parameter estimation in Section 2.4.2 and 2.4.2, and finally illustrate how to estimate each component in CompExp end-to-end in Section 2.4.2.

Extract-and-Refine Architecture

Our proposed model architecture for CompExp is shown in Figure 2.8, which in a nutshell is a fully connected hierarchical neural network. The explanations for a user item pair (u, c) is generated via an extract-and-refine process, formally described in Eq Eq (2.4.1). Comparing to existing pure generation-based explanation methods [37, 38, 116], one added benefit of our solution is to ensure faithfulness of the generated explanations: it avoids mentioning attributes that are not relevant to the target item. To address the limitations in directly using existing content, e.g., unaligned content style or sentiment polarity, the refinement step further rewrites the extracted sentence to make its content better fit for the purpose of comparative explanation, e.g., improve the quality defined by $\pi(x|u, c)$.

We refer to $P_{ext}(x_j^c|r_c^u, \Omega_u)$ as the extractor and $P_{ref}(x|x_j^c, r_c^u, \Omega_u)$ as the refiner. Next, we will zoom into each component to discuss its design principle and technical details.

Extractor The extractor's goal is to select a prototype sentence x_j^c from item c's profile Ψ_c for a given opinion rating r_c^u that best satisfies the comparativeness suggested by the user profile Ω_u. We refer to $x_j^c \in \Psi_c$ as an extraction candidate and $x_i^u \in \Omega_u$ as a reference. The extractor adopts a bidirectional GRU [123] as the universal text encoder to convert the extraction candidates and references into continuous embedding vectors. Since the pairwise comparison specified by Δr_i^u is a scalar, we use a one-hot vector to encode it when the ratings are discrete, otherwise we use a non-linear multi-layer perceptron (MLP) as the rating encoder.

Intuitively, in the one dimensional rating space, we can easily recover the intended sentence's rating r_c^u from the rating of the reference sentence r_i^u and required rating difference Δr_i^u. As an analogy, we consider the rating difference vector as the transform direction that suggests the ideal comparative explanation in the latent text space from a reference sentence x_i^u, denoted as $f(x_i^u, \Delta r_i^u) \rightarrow h_i$. As a result, h_i is the text embedding vector for the ideal comparative explanation. The extractor implements such a transformation using an MLP taking the concatenation of the text embedding and rating difference embedding vectors as input.

Given the desired comparative explanation h_i, the extraction candidates can be evaluated by their similarities towards h_i. This specifies a directional distribution $Q(x; h_i)$ centered on h_i in the latent text embedding space. Since cosine is a commonly used similarity metric for text embeddings, we formulate $Q(x; h_i)$ as a von

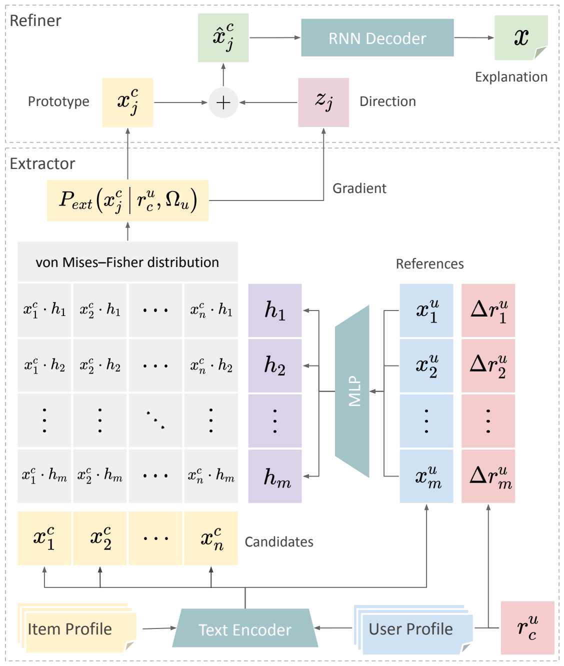


Figure 2.8: The extract-and-refine model architecture for CompExp. The extractor extracts a candidate sentence from item c's profile as a prototype for explanation generation; and the refiner rewrites this sentence to optimize the desired quality metric for comparative explanation.

Mises-Fisher distribution [141] over all the extraction candidates,

$$Q(x; h_i) \propto f_{vMF}(x; h_i, \kappa) = C_p(\kappa) e^{\kappa \cos(x, h_i)}$$

where $f_{vMF}(\cdot)$ is the probability density function, κ is the concentration parameter, and $C_p(\kappa)$ is a normalization function about k. Because each reference sentence x_i^u will suggest a different directional distribution, we extend the von Mises-Fisher distribution to cover multiple centriods and define $P_{ext}(x_j^c | r_c^u, \Omega_u)$ as follows,

$$P_{ext}(x_j^c | r_c^u, \Omega_u) \propto \sum_{x_i^u \in \Omega_u} f_{vMF}\Big(x_j^c; f(x_i^u, \Delta r_i^u), \kappa\Big) \tag{2.4.3}$$

Intuitively, in Eq Eq (2.4.3), each ideal embedding h_i suggests which extraction candidate better fits the comparativeness embedded in Ω_u. The summation over Ω_u aggregates each reference sentence's evaluation on candidate sentence x_j^c. κ is kept as a hyper-parameter which shapes the extraction probability distribution: a larger κ value leads to a skewer distribution. We can use it to control the exploration of the extraction candidates during the policy gradient based model training, which will be introduced in Section 2.4.2.

Refiner. The objective of the refiner is to rewrite the extracted prototype to further improve the quality metric $\pi(x|u, c)$. As we argued before, a better explanation should be more supportive to the pairwise comparison required by the user profile. Therefore, assuming the refiner successfully turns the prototype x_j^c into a better framed sentence $\widehat{x}_j^c$ about the item c for user u, then when we give $\widehat{x}_j^c$ back to the extractor together with x_j^c, the extractor should prefer the revised version over the original one. Otherwise, we should keep refining $\widehat{x}_j^c$ until the extractor believes it can no longer be improved. Hence, the refiner needs to find a direction such that $P_{ext}(x_j^c|r_c^u, \Omega_u) < P_{ext}(\widehat{x}_j^c|r_c^u, \Omega_u)$, which is exactly suggested by the gradient of $P_{ext}(x_j^c|r_c^u, \Omega_u)$ with respect to x_j^c, i.e., the fastest direction for x_j^c to increase the value of $P_{ext}(x_j^c|r_c^u, \Omega_u)$. As a result, our refiner simply pushes the text embedding vector of x_j^c alone this gradient direction:

$$\begin{aligned} z_j &= \nabla_{x_j^c} P_{ext}(x_j^c|r_c^u, \Omega_u) \\ &\propto \sum_i^m e^{\kappa \cos(x_j^c, h_i)} \Big[\frac{h_i}{|x_j^c||h_i|} - \cos(x_j^c, h_i)\frac{x_j^c}{|x_j^c|^2}\Big] \end{aligned}$$

Since the refinement step should only polish the extracted prototype instead of dramatically changing it, we normalize the gradient to a unit vector and restrict the step size to one in all cases, i.e., $\widehat{x}_j^c = x_j^c + z_j/|z_j|$. At last, we include a single-layer GRU with attention [146] as the text decoder to convert the refined text vector $\widehat{x}_j^c$ to the final explanation sentence x.

Connecting these two modules together, CompExp generates explanations for a ranked list of recommended items one at a time. To understand why the generated explanations carry comparativeness, we can consider the user's profile Ω_u as an anchor. Because all the explanations are generated against this anchor, the comparisons among the explanations emerge.

Explanation Quality Metric

To train CompExp under Eq Eq (2.4.2), we need to define the explanation quality metric $\pi(x|u,c)$. There is no commonly agreed offline metric for explanation quality in the community yet. And obtaining real user feedback is not feasible for offline model training. Currently, most of explainable recommendation solutions [37, 38, 116] adopt metrics measuring the overlapping content between the generated explanations and user reviews, such as BLEU [135].

However, the BLEU metric, which is initially designed for machine translation, is problematic in explanation evaluation for at least two important reasons. First, it is biased towards shorter sentences. As a precision-based metric, BLEU overcomes the short-length issue by introducing the brevity penalty, which down-scales the precision when the generated length is smaller than its "best match length" [135]. The "best match length" design is reasonable in machine translation, because all reference sentences are valid translations covering the information contained in the source language, regardless of their length differences. However, when using review sentences as proxies of explanations, the reference sentences from one review can describe totally different aspects of the same item and vary significantly in length and information contained. Since short-length generation benefits precision (less prone to erroneous word choices), BLEU favors explanations exploiting the short references as the "best match". As a result, it pushes the models to generate explanations that are generally much shorter than the average sentence length in a review, and hence fails to explain the item in details. Second, though precision-based, BLEU is incapable to differentiate the importance of different words in a reference sentence. Words are valued equally in machine translation, but their impact in explanations varies significantly to users. BLEU's indiscrimination to words unavoidably favors the explanations with more generic content due to their higher chance of appearance. We later demonstrate how the BLEU metric led to both short and generic explanations in our experiments.

To design a more appropriate metric to evaluate the explanation quality and better guide our model training, we propose IDF-BLEU, i.e., Inverse Document Frequency (IDF) enhanced BLEU. It introduces three changes on top of BLEU to balance the important factors in explanations: length, content overlapping, and content rarity.

First, to penalize an overly short generation, we replace the "best match length" in the brevity factor with the average length of sentences from all reviews,

$$BP_{len} = e^{\min(1-\frac{l_r}{l_x},0)}$$

where l_r and l_x is the average length of references and the length of the explanation respectively. Second, to differentiate the importance of different words, we introduce IDF to measure the value of n-grams and use it to reweigh the precision in BLEU. We compute the IDF of word g by the number of sentences where it occurs,

$$IDF(g) = log\frac{S}{s_g} + 1$$

where S is the total number of review sentences in the training corpus and s_g is the number of sentences containing word g. We approximate the IDF of an n-gram by the largest IDF of its constituent words. Then the clipped n-gram precision in BLEU is modified as

$$p_n = \frac{\sum_{g^n \in x} IDF(g^n) \cdot Count_{clip}(g^n)}{\sum_{g^n \in x} IDF(g^n) \cdot Count(g^n)} \tag{2.4.4}$$

where g^n represents the n-gram and $Count_{clip}(g^n)$ is the BLEU's operation to calculate the count of g^n in sentence x while being clipped by the corresponding maximum count in the references. Through the reweighing, correctly predicting an informative word becomes more rewarding than a generic word. However, it alone cannot evaluate content rarity, since the precision-based metric cannot punish sentences for not including rare words. Therefore, at last, inspired by the length brevity factor in original BLEU, we introduce a similar IDF brevity factor to punish sentences lacking words with high IDF,

$$BP_{IDF} = e^{\min(1-\frac{d_r}{d_x},0)}$$

where d_x is the average IDF per word $d_x = \sum_{g \in x} IDF(g)/l_x$ and d_r is corresponding average value in references. Then combining them forms our IDF-BLEU,

$$IDF - BLEU = BP_{len} \cdot BP_{IDF} \cdot \mathbb{E}\Big(\sum_{n=1}^{N} w_n \log p_n\Big) \tag{2.4.5}$$

where w_n is BLEU's parameter used as the weight of the n-gram precision. We use the proposed IDF-BLEU as the quality metric $\pi(x|u,c)$ for CompExp training.

Hierarchical Rewards

CompExp is a fully connected neural network which can be trained end-to-end with the gradient derived from Eq Eq (2.4.2). However, blind end-to-end training faces the risk that the model violates the purpose of our designed extract-and-refine procedure, as the model has a great degree of freedom to arbitrarily push the prototype x_j^c in the continuous vector space to optimize Eq Eq (2.4.2). For example, it could disregard the extracted prototype and generate totally irrelevant content to the target item c in the refiner.

To enforce the extract-and-refine workflow, we introduce additional intrinsic reward [147] for each layer respectively to regularize their behaviours. Specifically, as IDF-BLEU is used to measure the explanation quality in Eq Eq (2.4.2), we directly use the extracted sentence's IDF-BLEU to reward the extractor, i.e., introduce $\pi_{ext}(x_j^c|u,c) = IDF - BLEU(x_j^c)$. For the refiner, we discourage it in pushing the final generation too far away from the extracted one. Inspired by the clipped precision in Eq Eq (2.4.4), we propose a clipped recall to measure how many words from the selected sentence x_j^c are still covered in the refined sentence,

$$a_n = \frac{\sum_{g^n \in x_j^c} IDF(g^n) \cdot min[Count_{clip}(g^n), Count_x(g^n)]}{\sum_{g^n \in x_j^c} IDF(g^n) \cdot Count_{clip}(g^n)} \tag{2.4.6}$$

where $Count_{clip}(g^n)$ is the clipped count of n-gram g^n towards the references like in BLEU, and $Count_x(g^n)$ is the count of g^n in the refined explanation x. In other words, the denominator is the prototype's overlap with the target references and the numerator is the overlap among the prototype, references, and the final explanation. We did not use classical recall definition because it would reward the refiner to retain the entire prototype. We only encourage the refiner to keep the n-grams that are actually presented in the references. We compute the refiner's intrinsic reward by aggregating the clipped recall over different n-grams $\pi_{ref}(x, x_j^c) = \exp\big(\sum_{n=1}^{N} w_n \log a_n\big)$. We did not provide this reward to the extractor, because it biases the extractor to short and generic candidates which are easier for the refiner to cover.

With the hierarchical intrinsic rewards introduced for each component, we can optimize Eq Eq (2.4.2) by policy gradient as

$$\begin{aligned}\nabla_\Theta J \approx &[\lambda_1 \pi(x|u,c) + \lambda_2 \pi_{ref}(x, x_j^c)] \nabla_\Theta \log P_{ref}(x|x_j^c, r_c^u, \Omega_u) \\ &+ [\lambda_3 \pi(x|u,c) + \lambda_4 \pi_{ext}(x_j^c)] \nabla_\Theta \log P_{ext}(x_j^c|r_c^u, \Omega_u)\end{aligned}$$

where λ_1 to λ_4 are coefficients to adjust the importance of each reward, and Θ stands for the model parameters in CompExp.

Table 2.12: Summary of the processed datasets.

Dataset	# Users	# Items	# Reviews	Rating Range
RateBeer	6,566	19,876	2,236,278	0 - 20
TripAdvisor	4,954	4,493	287,879	1 - 5

Model Training

The whole model training process can be organized into two steps: pre-training and fine-tuning. The pre-training step aims to bootstrap the extractor and refiner independently. To prepare the extractor to recognize the comparative relationships among sentences, we treat every observed review sentence as the extraction target and train the extractor to maximize its negative log-likelihood with regard to the corresponding user and item profiles.

It is important to pre-train the refiner as a generative language model, because it would be very inefficient to learn all the natural language model parameters only through the end-to-end training. However, we do not have any paired sentences to pre-train the refiner. We borrowed the method introduced in [138, 141] to manually craft such pairs. Specifically, for every sentence, we compute its cosine similarity against all other sentences in the same item profile in the latent embedding space, and select the most similar one to pair with. Then we use this dataset to pre-train the refiner with negative log-likelihood loss.

In the fine-tuning stage, we concatenate the pre-trained layers and conduct the end-to-end training with policy gradient. To make the policy gradient training more resilient to variance and converge faster, it is important to have a baseline to update the model with reward advantages instead of using the rewards directly. We apply Monte Carlo sampling in both extractor and refiner to have multiple explanations, and use their mean rewards as the baseline.

2.4.3 Experiments

We demonstrate empirically that CompExp can generate improved explanations compared to state-of-the-art explainable recommendation algorithms. We conduct experiments on two different recommendation scenarios: RateBeer reviews with single-ratings [130] and TripAdvisor reviews with *multi-aspect ratings*.

Experiment Setup. As our solution only focuses on explanation generation, it can be applied to any recommendation algorithm of choice. In our experiments, we directly use the ground-truth review ratings as the recommendation score to factor out any deviation or noise introduced by specific recommendation algorithms. For completeness, we also empirically studied the impact from input ratings if switched to a real recommendation algorithm's predictions.

Data Pre-Processing. In the RateBeer dataset, we segment each review into sentences, and label them with the overall ratings from their original reviews. In the TripAdvisor dataset, there are separate ratings for five aspects including *service, room, location, value and cleanliness*. Therefore, each TripAdvisor review is expected to be a mix of a user's opinions on these different aspects about the item. We segment sentences in a TripAdvisor review to different aspects using the boot-strapping method from and assign resulting sentences the corresponding aspect ratings. These two datasets evaluate CompExp under different scenarios: overall opinion vs., aspect-specific opinion. We also adopt the recursive filtering [108] to alleviate the data sparsity. The statistics of the processed datasets are summarized in Table 2.12.

Baselines We compared with three explainable recommendation baselines that generate natural language explanations, covering both extraction-based and generation-based solutions.

- **NARRE**: Neural Attentional Regression model with Review-level Explanations [110]. It is an extraction-based solution. It learns the usefulness of the existing reviews through attention and selects the most attentive reviews as the explanation.
- **NRT**: Neural Rating and Tips Generation [37]. It is a generation-based solution. It models rating regression and content generation as a multi-task learning problem with shared latent space. Content is generated from its neural language model component.

Table 2.13: Explanation quality evaluated under IDF-BLEU, BLEU, average sentence length, average IDF per word, rep/l, seq_rep_2, feature precision and recall on RateBeer and TripAdvisor datasets. Bold numbers are the best of the corresponding metrics with p-value < 0.05.

Model	IDF-BLEU 1	IDF-BLEU 2	IDF-BLEU 4	BLEU 1	BLEU 2	BLEU 4	Avg Length	IDF/word	Feature precision	Feature recall
RateBeer										
Human	/	/	/	/	/	/	11.13	2.45	/	/
NARRE	17.00	5.18	1.29	30.22	9.90	3.58	11.50	2.43	0.2217	0.0722
NRT	30.38	16.30	5.80	48.22	25.28	10.03	10.43	2.09	0.4563	0.1320
SAER	31.79	16.02	5.71	49.08	26.87	10.59	10.71	1.93	0.4751	0.1347
CompExp	**32.36**	**19.55**	**6.95**	49.14	29.63	11.41	10.52	2.16	**0.4796**	**0.1383**
TripAdvisor										
Human	/	/	/	/	/	/	12.85	2.45	/	/
NARRE	11.97	3.43	1.59	20.45	6.23	3.38	13.17	2.41	0.1733	0.1258
NRT	16.19	7.50	**2.48**	30.62	13.07	5.11	10.22	1.81	0.2939	0.1866
SAER	16.37	7.65	2.35	31.20	13.51	4.94	10.08	1.71	**0.3178**	**0.1961**
CompExp	**21.35**	**8.01**	2.16	31.70	12.23	4.16	13.35	2.12	0.3155	0.1930

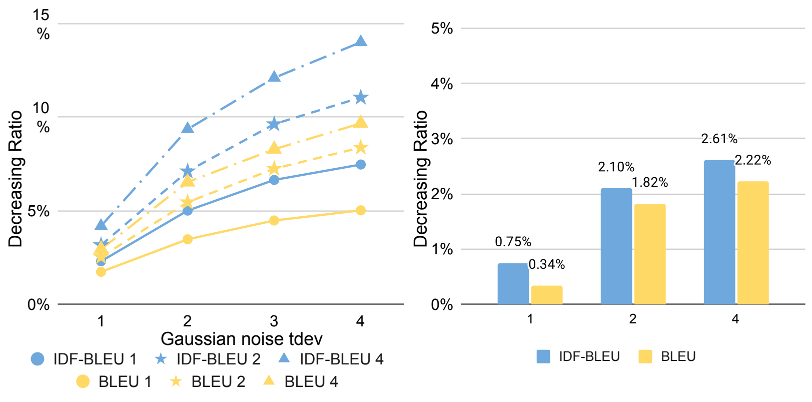
- **SAER**: Sentiment Aligned Explainable Recommendation [38]. This is another generation-based solution using multi-task learning to model rating regression and explanation generation. But it focuses specifically on the sentiment alignment between the predicted rating and generated explanation.

2.4.4 Quality of Generated Explanations

To comprehensively study the quality of generated explanations, we employ different types of performance metrics, including IDF-BLEU-{1, 2, 4}, BLEU-{1, 2, 4}, average sentence length, average IDF per word, and feature precision & recall. Features are items' representative attributes that users usually care the most [38,108]. The precision and recall measure if features mentioned in the generated explanations also appear in the user's ground-truth review. We also include ground-truth review sentences as a reference baseline (labeled as "Human") to study the differences between human and algorithm generated content. The results are reported in Table 2.13.

Advantages of CompExp. There is clear performance gap between the extraction-based solutions (NARRE) and generation-based ones (NRT, SAER, CompExp). While generation-based solutions largely outperformed extraction-based ones in content overlapping with ground-truth (IDF-BLEU, BLEU, feature precision and recall), they were generally very different from human writings in terms of sentence length, use of rare words (IDF/word). The extraction-based solutions use content provided by human, but they are limited to the existing content. The generation-based solutions customize content for each recommendation, but suffer from common flaws of generative models, e.g., short, dull, and repetitive. Among all the models, CompExp achieved the best balance among all metrics. It significantly exceeded all baselines in terms of IDF-BLEU-{1,2}. Its feature precision and recall are competitive with SAER while leading the rest, though SAER enjoys additional advantage from predefined feature pool of each item as input. As a generation-based model, CompExp largely improved the average length, word rarity, and reduced repetition over NRT and SAER.

Comparativeness. To verify if the generated explanations by CompExp capture the comparative ranking of items, we study its its output's sensitivity to the input recommendation ratings. As a starting point, the ground-truth explanation perfectly aligns with the recommendation ranking, which is derived from the ground-truth rating. If the generated explanation carries the same ranking of item, the generated content should be close to the ground-truth content. As a result, if we manipulate the input recommendation scores of items, the generated explanations should start to deviate. The further we push the rankings apart, the further the generated explanation should be pushed away from the ground-truth explanation. We use IDF-BLEU and BLEU to measure the content similarity and perturb the recommendation ratings with Gaussian noise. As shown in Figure 2.9a, all IDF-BLEU and BLEU metrics keep decreasing with the increasing amount of perturbation. In other words, even if it is for the same user and same set of items, with different recommendation scores assigned, CompExp would generate different explanations to explain their relative ranking.



(a) (b)

Figure 2.9: (a) Impact of noise in recommendation ratings on BLEU and IDF-BLEU. (b) Change in BLEU and IDF-BLEU with algorithm's predicted ratings.

Predicted Ratings. Motivated by the findings in Figure 2.9a, we further study how CompExp is influenced by a real recommendation algorithm's predicted ratings. We employed the neural collaborative filtering [27] and used its predicted ratings in CompExp's training and testing. The result is plotted in Figure 2.9b. Compared with previous randomly perturbed ratings, the predicted ratings bring very limited changes to the explanations. This confirms our experiment results based on ground-truth ratings can fairly represent CompExp's performance in real-world usage scenarios.

2.4.5 User Study

We have three research questions to answer in user study: 1) does users' judgement toward explanation quality aligns more with IDF-BLEU than BLEU; 2) do users find our comparative explanations more helpful than the baselines'; and 3) can users better perceive the comparative ranking from our explanations than the baselines'. To answer these three research questions, we design two user study tasks based on RateBeer dataset using Amazon Mechanical Turk.

The first task studies the first two research questions together. Specifically, we shuffle explanations from different models about the same recommended item and ask the participants to compare them, and then select the most helpful ones. To help participants evaluate the explanation quality, we include the original user review as the item description, towards which they can judge if the explanation are accurate or informative. For each recommended item, we ask participants to answer the following question after reading its description and candidate explanations:

> *"Which of the following explanations best describe the characteristics of the given beer and help you the most to understand why you should pay attention to the recommendation?"*

In this experiment, we collected 660 user responses.

The results are presented in Table 2.14 and 2.15. In Table 2.14, we used Cohen's kappa coefficient to compare IDF-BLEU and BLEU's agreement with users' responses. For each test case, we pair explanations that the participants chose as helpful with the rest to form a set of explanation pairs. Then we use IDF-BLEU-{1,2,4} and BLEU-{1,2,4} to identify the helpful one in each pair. The kappa coefficient shows that IDF-BLEU aligns significantly better with users' judgment in all three subcategories under paired t-test. Table 2.15 shows the helpfulness vote on each model and the paired t-test results of CompExp against other baselines. The helpfulness vote on CompExp is significantly higher than others, which suggests strong user preference over its generated explanations.

The second task addresses the last research question, i.e., if a user is able to perceive the ranking of recommended items from the explanations. In this task, we randomly paired items of different ratings and asked participants to identify which item is better by reading the provided explanations. We then evaluated the

Table 2.14: Cohen's kappa coefficient of explanation quality between the human judgements and BLEU & IDF-BLEU.

		1	2	4
κ	BLEU	0.2936	0.3114	0.2814
	IDF-BLEU	0.3452	0.3396	0.3152
Paired t-test		0.0001	0.0094	0.0071

Table 2.15: Up-vote rate of explanations' helpfulness.

	CompExp	SAER	NRT	NARRE
Up-vote Rate	43.79%	37.27%	35.61%	30.61%
Paired t-test	/	0.0182	0.009	0

agreement rate between participants' choices and the actual ranking. In particular, given the explanations of a model, the participants were required to answer the following question:

> *"After reading the explanations for recommended items, which item would you like to choose? You are expected to judge the quality of the items based on the provided explanations."*

We chose SAER and NRT as baselines. Besides, we also include the ground-truth sentences from the actual user reviews as a reference. We collected 200 responses for each model.

Table 2.16 reports the agreement rates between the actual ranking and the ranking perceived by the participants. CompExp's agreement rate is slightly higher than NRT and SAER, but it is far below the Ground-Truth. The Ground-Truth's high agreement rate quantitatively confirms that the original user provided review sentences are highly comparative. This observation supports our choice of training the comparative explanation generation from paired user review sentences. And it also suggests there is still a performance gap in comparativeness for learning-based solutions to bridge. And an improved objective for optimization, e.g., include quantified pairwise comparativeness, might be a promising direction.

2.5 Conclusion

This chapter is dedicated to enhancing the transparency of PS by providing personalized, intuitive textual explanations that are tailored to users' preferences. Our objective is to make these explanations informative, faithful, readable, and comparable. To achieve this, we present several techniques. First, we introduce MTER, a method that mines the rich opinionated content in user reviews to generate informative, opinionated explanations. To improve the fidelity of explanations, we propose FacT, which integrates a rule-based decision tree into latent factor models to learn explainable user and item representations. In addition to template-based explanation generation, we also explore natural language explanation generation, which offers greater expressiveness and diversity. To ensure the sentiment of the explanations aligns with the recommendation results, we develop SAER. Then we also introduce CompExp, a method to make explanations more comparable to better assist users' decision-making process. Our extensive experiments demonstrate the effectiveness of our methods, and serious user studies confirm the practical value of the explanations generated by our approaches.

We primarily focused on the generation of explanations with the goal of helping users make more informed-decisions and improving their trust in the system. However, there are many other aspects need to be considered in serving explanations to improve PS's transparency. Firstly, the explanations are personalized towards the target users/items, fairness issue make arise if the explanations associate to the sensitive attributes of users/items being served. Second, the evaluation of explanations is a fundamental problem, since it is hard to measure how users will react to the explanations unless we can do online experiments or simulated user study, which can be quite expensive. Third, instead of generating static explanations, dynamic explanations that interact with user behaviors can also be an interesting avenue to explore. This can provide users with real-time feedback and help them better understand how the system is making recommendations. It can also help improve the system's performance by learning from users' behaviors and preferences. Overall, there are many different aspects to consider when it comes to serving explanations to improve transparency in PS. It's important to continue exploring different techniques and approaches to ensure that explanations are accurate, trustworthy, and beneficial to users.

Table 2.16: Agreement rate between actual ranking and the users perceived ranking of paired items based on the provided explanations.

	GT	CompExp	SAER	NRT
Agreement Rate	72.29%	57.27%	56.25%	53.14%

Chapter 3

Promoting Fairness in Personalized Results

Personalization systems (PS) have traditionally been celebrated for their ability to assist users and increase business revenue. However, recent concerns from academia and industry have highlighted the potential for PS to perpetuate bias and unfairness. These issues of unfairness have far-reaching implications, often leading to negative consequences for underrepresented or disadvantaged groups. For example, an e-commerce systems may promote items that maximize profits for certain business owners, which can disadvantage smaller businesses. In online job marketplaces, recommender systems may perpetuate racial or gender-based discrimination by disproportionately recommending low-paying jobs to certain user groups. Therefore, it is imperative that we examine fairness in PS and develop systems that are built on trust and promote equality. By addressing these issues, we can work towards creating a more inclusive and equitable society, while also improving the satisfaction of all stakeholders involved in PS.

In this chapter, our main focus is on promoting fairness in PS from different perspectives. Various fairness definitions have been proposed and applied to different parts of PS. We center our attention on two types of fairness: universal fairness and measure-specific fairness. We develop general frameworks to address these types of fairness. In terms of universal fairness, our objective is to learn user or item representations that are independent of sensitive attributes to prevent bias or fairness issues when they are applied in downstream prediction tasks. By doing so, we can ensure that sensitive attributes do not influence the personalized results made by the PS. Regarding measure-specific fairness, we examine potential bias and fairness issues in generated natural language explanations that are used to serve users and recommend items. We propose specific fairness notions and techniques to tackle these concerns.

3.1 Learning Unbiased Representations from Biased Graph Observations

Graph embedding is an indispensable building block in modern machine learning approaches [57–61]. Graph embedding methods map each node to a low-dimensional embedding vector that reflects the nodes' structural information from the observed connections in the given graph. These node embeddings are then employed to solve downstream tasks, such as friend recommendation in social networks or user interest prediction in e-commerce platforms [62, 63].

However, the observed node connections in a graph are inevitably affected by certain *sensitive node attributes* (e.g., gender, age, race, religion, etc., of users) [148], which are intended to be withheld from many high-stake real-world applications. Without proper intervention, the learned node embeddings can inherit undesired sensitive information and lead to severe bias and fairness concerns in downstream tasks [17, 149]. For example, in social network recommendation, if the users with the same gender are observed to connect more often, the learned embeddings can record such information and lead to gender bias by only recommending friends to a user with the same gender identity. Biased node embeddings, when applied in applications such as loan application [150] or criminal justice [151], may unintentionally favor or disregard one demographic group, causing unfair treatments. These realistic and ethical concerns set a higher bar for the graph embedding methods to learn both effective and unbiased embeddings.

To date, the most popular recipe for unbiased graph embedding is to add adversarial regularizations to the loss function, such that the sensitive attributes cannot be predicted from the learned embeddings [149, 152–154]. However, such a regularization is only *a necessary condition* for debiasing node embeddings, and it usually hurts the utility of the embeddings (a trivial satisfying solution is to randomize the embeddings). Besides these regularization-based solutions, Fairwalk [17] modifies the random walk strategy in the node2vec algorithm [59] into two levels: when choosing the next node on a path, it first randomly selects a group defined by sensitive attributes, and then randomly samples a reachable node from that group. DeBayes [155] proposes to capture the sensitive information by a prior function in Conditional Network Embedding [156], such that the learned embeddings will not carry the sensitive information. Nevertheless, both Fairwalk and DeBayes are based on specific graph embedding methods; and how to generalize them to other types of graph embedding methods such as GAT [157] or SGC [158] is not obvious.

Moving beyond the existing unbiased graph embedding paradigm, in this work, we propose a principled new framework for the purpose with theoretical justifications. Our solution is to learn node embeddings from an *underlying bias-free graph* whose edges are generated without influence from sensitive attributes. Specifically, as suggested by Pfeiffer et al. [148], the generation of a graph can be treated as a two-phase procedure. In the first phase, the nodes are connected with each other solely based on global graph *structural properties*, such as degree distributions, diameter, edge connectivity, clustering coefficients and etc., resulting in an *underlying structural graph*, free of influences from node attributes. In the second phase, the connections are *re-routed* by the node attributes (including both sensitive and non-sensitive attributes). Hence, our debiasing principle is to filter out the influence from sensitive attributes on the underlying structural graph to create a bias-free graph (that only has non-sensitive or no attributes) from the observed graph, and then perform embedding learning on the bias-free graph.

We propose two alternative ways to uncover the bias-free graph from the given graph for learning node embeddings. The first is a weighting-based method, which reweighs the graph reconstruction based loss function with importance sampling on each edge, such that the derived loss is as calculated on the bias-free graph, in expectation. This forms *a sufficient condition* for learning unbiased node embeddings: when the reconstruction loss is indeed defined on the corresponding bias-free graph, the resulting node embeddings are unbiased, since the bias-free graph is independent from the sensitive attributes. The second way is via regularization, in which we require that, with and without the sensitive attributes, the probabilities of generating an edge between two nodes from their embeddings are the same. In contrast, this forms *a necessary condition*: when the learning happens on the bias-free graph, the resulting embeddings should not differentiate if any sensitive attributes participated in the generation of observed graph, i.e., the predicted edge generation should be independent from the sensitive attributes. These two methods are complementary and can be combined to control the trade-off between utility and unbiasedness.

Comprehensive experiments on three datasets and several backbone graph embedding models prove the effectiveness of our proposed framework[1]. Results also suggest that the embeddings from our methods can lead to *fair* predictions in the downstream applications.

3.1.1 Related Work

Graph embedding aims to map graph nodes to low-dimensional vector representations such that the original graph can be reconstructed from these node embeddings. Traditional approaches include matrix factorization and spectral clustering techniques [159, 160]. Recent years have witnessed numerous successful advances in deep neural architectures for learning node embeddings. Deepwalk [57] and node2vec [59] utilize a skip-gram [161] based objective to recover the node context in random walks on a graph. Graph Convolutional Networks (GCNs) learn a node's embedding by aggregating the features from its neighbors supervised

[1]Our code: `https://github.com/MyTHWN/UGE-Unbiased-Graph-Embedding`.

by node/edge labels in an end-to-end manner. These techniques are widely applied in friend or content recommendation [162, 163], protein structure prediction [164], and many more.

Recent efforts on unbiased and fair graph embedding mainly focus on *pre-processing*, *algorithmic* and *post-processing* steps in the learning pipeline. The pre-processing solutions modify the training data to reduce the leakage of sensitive attributes [16]. Fairwalk [17] is a typical pre-processing method which modifies the sampling process of random walk on graphs by giving each group of neighboring nodes an equal chance to be chosen. However, such pre-processing may well shift the data distribution and leads the trained model to inferior accuracy and fairness measures. The *post-processing* methods employ discriminators to correct the learned embeddings to satisfy specific fairness constraints [24]. However, such ad-hoc post-correction is detached from model training which can heavily degrade model's prediction quality.

Our work falls into the category of *algorithmic* methods, which modify the learning objective to prevent bias from the node embeddings. The most popular algorithmic solution is adding (adversarial) regularizations as constraints to filter out sensitive information [149, 153, 165]. Compositional fairness constraints [149] are realized by a composition of discriminators for a set of sensitive attributes jointly trained with the graph embedding model. Similarly, FairGNN [165] adopts a fair discriminator but focuses on debiasing with missing sensitive attribute values. Different from regularization based methods. DeBayes [155] reformulates the maximum likelihood estimation with a biased prior which absorbs the information about sensitive attributes; but this solution is heavily coupled with the specific embedding method thus is hard to generalize. Our method differs from these previous works by learning embeddings from an underlying bias-free graph. We investigate the generation of the given graph and remove the influence from sensitive attributes in the generative process to uncover a bias-free graph for graph embedding.

3.1.2 Preliminaries

In this section, we first introduce our notations and general graph embedding concepts. Since the bias/fairness issues emerge most notably in prediction tasks involving humans, such as loan application or job recommendation, we will use user-related graphs as running examples to discuss our criterion for unbiased graph embedding. But we have to emphasize that this setting is only to illustrate the concept of unbiased graph embedding; and our proposed solution can be applied to any graph data and selected sensitive attributes to avoid biases in the learned embeddings.

Notation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$ be an undirected, attributed graph with a set of N nodes $\mathcal{V}$, a set of edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, and a set of N attribute vectors $\mathcal{A}$ (one attribute vector for each node). We use (u, v) to denote an edge between node u and node v. The number of attributes on each node is K, and $\mathcal{A} = \{\boldsymbol{a}_1, \boldsymbol{a}_2, \dots, \boldsymbol{a}_N\}$, where $\boldsymbol{a}_u$ is a K-dimensional attribute value vector for node u. We assume all attributes are categorical and $\mathcal{S}_i$ is the set of all possible values for attribute i. [2] For example, if node u is a user node, and the i-th attribute is gender with possible values $\mathcal{S}_i = \{\textit{Female}, \textit{Male}, \textit{Unknown}\}$, then $\boldsymbol{a}_u[i] = \textit{Female}$ indicates u is a female. Without loss of generality, we assume the first m attributes are sensitive, and $a_u[: m]$ and $a_u[m :]$ stands for the m sensitive attributes and the rest of the attributes that are non-sensitive, respectively.

In the problem of graph embedding learning, we aim to learn an encoder $\mathrm{ENC} : \mathcal{V} \to \mathbb{R}^d$ that maps each node u to a d-dimensional embedding vector $\boldsymbol{z}_u = \mathrm{ENC}(u)$. We focus on the *unsupervised* embedding setting which does not require node labels and the embeddings are learned via the *link prediction task*. In this task, a scoring function $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ with parameters $\boldsymbol{\theta}$ is defined to predict the probability of an edge $(u, v) \in \mathcal{E}$ between node u and node v in the given graph. The loss for learning node embeddings and parameters of the encoder and scoring function is defined by:

$$\sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)), \tag{3.1.1}$$

[2] We acknowledge that there are cases where attribute values are continuous, where discretization techniques can be applied.

where $\mathcal{L}_{edge}$ is a per-edge loss function on $(u, v) \in \mathcal{E}$. Such loss functions generally aim to maximize the likelihood of observed edges in the given graph, comparing to the negative samples of node pairs where edges are not observed [59, 166].

Unbiased Graph Embedding

Given a node u, we consider its embedding $\boldsymbol{z}_u$ as unbiased with respect to an attribute i if it is independent from the attribute. Prior works evaluate such unbiasedness in the learned node embeddings by their ability to predict the values of the sensitive attributes [149, 155, 167]. For example, they first train a classifier on a subset of node embeddings using their associated sensitive attribute values as labels. If the classifier cannot correctly predict the sensitive attribute values on the rest of node embeddings, one claims that the embeddings have low bias. If the prediction performance equals to that from random node embeddings, the learned embeddings are considered bias-free. In fact, such classifiers are often used as discriminators in adversarial methods where the classifier and the embeddings are learned jointly: the embeddings are pushed in directions where the classifier has low prediction accuracy [149, 152].

There are also studies that use fairness measures such as demographic parity or equalized opportunity to define the unbiasedness of learned embeddings [24, 155]. But we need to clarify that such fairness measures can only evaluate the fairness of the final prediction results for the intended downstream tasks, but cannot assess whether the embeddings are biased by, or contain any information about, sensitive attributes. In particular, fairness in a downstream task is only a necessary condition for unbiased embedding learning, not sufficient. The logic is obvious: unbiased embeddings can lead to fair prediction results as no sensitive attribute information is involved; but obtaining fairness in one task does not suggest the embeddings themselves are unbiased, e.g., those embeddings can still lead to unfair results in other tasks or even the fair results are obtained by other means, such as post-processing of the prediction results [168]. In Section 3.1.5, we will use both the prediction accuracy on sensitive attributes and fairness measures on final tasks to evaluate the effectiveness of our unbiased graph embedding methods.

3.1.3 Effect of attributes in graph generation

In this section, we discuss the generation of an observed graph by explicitly modeling the effects of node attributes in the process. In particular, we assume that there is an *underlying structural graph* behind an observed graph, whose edge distribution is governed by the global graph structural properties such as degree distributions, diameter, and clustering coefficients. The attributes in $\mathcal{A}$ will modify the structural edge distribution based on effects like *homophily* in social networks, where links are rewired based on the attribute similarities of the individuals [169, 170]. The modified edge distribution is then used to generate the observed graph.

Formally, let $\mathcal{M}$ be a structural generative graph model and Θ_M be the set of parameters that describe properties of the underlying structural graph. In particular, this set of parameters Θ_M is independent from node attributes in $\mathcal{A}$. We consider the class of models that represent the set of possible edges in the graph as binary random variables $E_{uv}, u \in \mathcal{V}, v \in \mathcal{V}$: i.e., the event $E_{uv} = 1$ indicates $(u, v) \in \mathcal{E}$. The model $\mathcal{M}$ assigns a probability to E_{uv} based on Θ_M, $P_M(E_{uv} = 1|\Theta_M)$. Therefore, the edges of an underlying structural graph $\mathcal{G}_M$ can be considered as samples from $Bernoulli(P_M(E_{uv} = 1|\Theta_M))$. There are many such structural models $\mathcal{M}$ such as the Chung Lu model [171] and Kronecker Product Graph Model [172]. Note that $\mathcal{M}$ does not consider node attributes in the generation of the structural graph.

Now we involve the attributes in the generative process. Let $C \in \{(\boldsymbol{a}_i, \boldsymbol{a}_j)| i \in \mathcal{V}, j \in \mathcal{V}\}$ be a variable indicating the *attribute value combination* of a randomly sampled pair of nodes, which is independent from Θ_M. Note that C instantiated by different node pairs can be the same, as different nodes can have the same attribute values. The conditional probability of an edge between u and v, given the corresponding attribute values on them and the structural parameters Θ_M, is $P_o(E_{uv} = 1|C = \boldsymbol{a}_{uv}, \Theta_M)$, where $\boldsymbol{a}_{uv} = (\boldsymbol{a}_u, \boldsymbol{a}_v)$

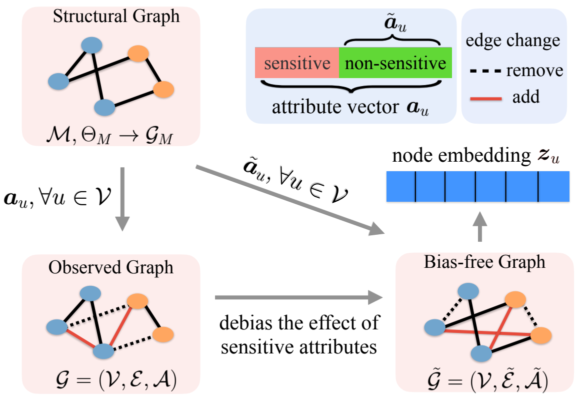

Structural Graph
$\mathcal{M}, \Theta_M \rightarrow \mathcal{G}_M$
$\tilde{\boldsymbol{a}}_u$
sensitive
non-sensitive
attribute vector $\boldsymbol{a}_u$
edge change
remove
add
node embedding $\boldsymbol{z}_u$
$\tilde{\boldsymbol{a}}_u, \forall u \in \mathcal{V}$
$\boldsymbol{a}_u, \forall u \in \mathcal{V}$
Observed Graph
$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{A})$
debias the effect of sensitive attributes
Bias-free Graph
$\tilde{\mathcal{G}} = (\mathcal{V}, \tilde{\mathcal{E}}, \tilde{\mathcal{A}})$

Figure 3.1: Illustration of Unbiased Graph Embedding (UGE). The color of the nodes represents the value of their attributes, and different line styles suggest how the observed edges are influenced by attributes in the generative process.

denotes the attribute value combination on nodes u and v. Based on Bayes' Theorem, we have

$$\begin{aligned}
&P_o(E_{uv} = 1 | C = \boldsymbol{a}_{uv}, \Theta_M) \\
=&\frac{P_o(C = \boldsymbol{a}_{uv} | E_{uv} = 1, \Theta_M) P_o(E_{uv} = 1 | \Theta_M)}{P_o(C = \boldsymbol{a}_{uv} | \Theta_M)} \\
=&P_M(E_{uv} = 1 | \Theta_M) \frac{P_o(C = \boldsymbol{a}_{uv} | E_{uv} = 1, \Theta_M)}{P_o(C = \boldsymbol{a}_{uv} | \Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V},
\end{aligned} \tag{3.1.2}$$

where the prior distribution on E_{uv} is specified by the structural model $\mathcal{M}$: i.e., $P_o(E_{uv} = 1|\Theta_M) = P_M(E_{uv} = 1|\Theta_M)$, and the posterior distribution accounts for the influences from the attribute value combinations. Therefore, the edge probabilities used to generate the observed graph with node attributes is a *modification* of those from a structural graph defined by $\mathcal{M}$ and Θ_M. It is important to clarify that the node attributes are given ahead of graph generation. They are the input to the generative process, not the output. Hence, $P_o(C = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)$ represents the probability that in all edges, the specific attribute value combination $\boldsymbol{a}_{uv}$ is observed on an edge's incident nodes. It is thus the same for all edges whose incident nodes have the same attribute value combination.

To simplify the notation, let us define a function that maps the attribute value combination $\boldsymbol{a}_{uv}$ to the probability ratio that modifies the structural graph into the observed graph by

$$R(\boldsymbol{a}_{uv}) := \frac{P_o(C = \boldsymbol{a}_{uv} | E_{uv} = 1, \Theta_M)}{P_o(C = \boldsymbol{a}_{uv} | \Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V}.$$

Thus we can rewrite Eq (3.1.2) by

$$P_o(E_{uv} = 1 | C = \boldsymbol{a}_{uv}, \Theta_M) = P_M(E_{uv} = 1 | \Theta_M) R(\boldsymbol{a}_{uv}). \tag{3.1.3}$$

In this way, we explicitly model the effect of node attributes by $R(\boldsymbol{a}_{uv})$, which modifies the structural graph distribution $P_M(E_{uv} = 1|\Theta_M)$ for generating the observed graph $\mathcal{G}$.

3.1.4 Unbiased Graph Embedding from a Bias-Free Graph

In this section, we describe our proposed methods for learning unbiased node embeddings based on the generative modeling of the effects of sensitive attributes in Section 3.1.3. In a nutshell, we aim to get rid of the sensitive attributes and modify the structural edge probabilities by only conditioning on non-sensitive attributes. This gives us the edge probabilities of a bias-free graph, from which we can learn unbiased node embeddings. We illustrate this principle in Figure 3.1. Consider a world without the sensitive attributes, and the attribute vector of node u becomes $\widetilde{\boldsymbol{a}}_u = \boldsymbol{a}_u[m :]$, which only include non-sensitive attributes in $\boldsymbol{a}_u$. We denote $\widetilde{\mathcal{G}} = (\mathcal{V}, \widetilde{\mathcal{E}}, \widetilde{\mathcal{A}})$ as the corresponding new graph generated with $\widetilde{\boldsymbol{a}}_u, \forall u \in \mathcal{V}$, and $\widetilde{\boldsymbol{a}}_{uv} = (\widetilde{\boldsymbol{a}}_u, \widetilde{\boldsymbol{a}}_v)$. Therefore,

$\widetilde{\mathcal{G}}$ is a bias-free graph without influence from sensitive attributes. If we can learn node embeddings from $\widetilde{\mathcal{G}}$ instead of $\mathcal{G}$, the embeddings are guaranteed to be unbiased with respect to sensitive attributes. Specifically, the edge probabilities used for generating $\widetilde{\mathcal{G}}$ can be written as

$$P_{\widetilde{o}}(E_{uv} = 1|\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M) = P_M(E_{uv} = 1|\Theta_M)\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}), \tag{3.1.4}$$

where

$$\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}) \coloneqq \frac{P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|E_{uv} = 1, \Theta_M)}{P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|\Theta_M)}, \forall u \in \mathcal{V}, \forall v \in \mathcal{V}, \tag{3.1.5}$$

$\widetilde{C} \in \{(\widetilde{\boldsymbol{a}}_i, \widetilde{\boldsymbol{a}}_j)|i \in \mathcal{V}, j \in \mathcal{V}\}$ is the random variable for attribute value combinations without sensitive attributes, and $P_{\widetilde{o}}$ indicates the distributions used in generating $\widetilde{\mathcal{G}}$. We name the class of methods that learn embeddings from $\widetilde{\mathcal{G}}$ as UGE, simply for **U**nbiased **G**raph **E**mbedding. Next we introduce two instances of UGE. The first is UGE-W, which reweighs the per-edge loss such that the total loss is from $\widetilde{\mathcal{G}}$ in expectation. The second method is UGE-R, which adds a regularization term to shape the embeddings to satisfy the properties as those directly learned from $\widetilde{\mathcal{G}}$.

Weighting-Based UGE

To compose a loss based on $\widetilde{\mathcal{G}}$, we modify the loss function in Eq (3.1.1) by reweighing the loss term on each edge as

$$\mathcal{L}_{UGE-W}(\mathcal{G}) = \sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}. \tag{3.1.6}$$

The following theorem shows that, in expectation, this new loss is equivalent to the loss for learning node embeddings from $\widetilde{\mathcal{G}}$.

Theorem 3.1.1. Given a graph $\mathcal{G}$, and $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})/R(\boldsymbol{a}_{uv}), \forall(u,v) \in \mathcal{E}$, $\mathcal{L}_{UGE-W}(\mathcal{G})$ is an unbiased loss with respect to $\widetilde{\mathcal{G}}$.

Proof. We take expectation over the edge observations in $\mathcal{G}$ as

$$\begin{aligned}
&\mathbb{E}\big[\mathcal{L}_{UGE-W}(\mathcal{G})\big] \\
&=\mathbb{E}\left[\sum_{(u,v)\in\mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})}\right] \\
&=\mathbb{E}\left[\sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})} \cdot E_{uv}\right] \\
&= \sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})} \cdot P_o(E_{uv} = 1|C = \boldsymbol{a}_{uv}, \Theta_M) \\
*&= \sum_{u\in\mathcal{V},v\in\mathcal{V}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \cdot P_{\widetilde{o}}(E_{uv} = 1|\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M) \\
&=\mathbb{E}\left[\sum_{(u,v)\in\widetilde{\mathcal{E}}} \mathcal{L}_{edge}(\mathrm{s}(\boldsymbol{z}_u, \boldsymbol{z}_v))\right].
\end{aligned} \tag{3.1.7}$$

The step marked by $*$ uses Eq (3.1.3) and Eq (3.1.4). □

UGE-W is closely related to the idea of importance sampling [173], which analyzes the edge distribution of the bias-free graph $\widetilde{\mathcal{G}}$ by observations from the given graph $\mathcal{G}$. The only thing needed for deploying UGE-W in existing graph embedding methods is to calculate the weights $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})/R(\boldsymbol{a}_{uv})$. To estimate $R(\boldsymbol{a}_{uv})$, we need the estimates of $P_o(C = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M)$ and $P_o(C = \boldsymbol{a}_{uv}|\Theta_M)$. With maximum likelihood estimates on the observed graph, we have

$$P_o(C = \boldsymbol{a}_{uv}|E_{uv} = 1, \Theta_M) \approx \frac{\sum_{(i,j)\in\mathcal{E}} \mathbb{I}[\boldsymbol{a}_{ij} = \boldsymbol{a}_{uv}]}{|\mathcal{E}|}, \tag{3.1.8}$$

$$P_o(C = \boldsymbol{a}_{uv}|\Theta_M) \approx \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V}} \mathbb{I}[\boldsymbol{a}_{ij} = \boldsymbol{a}_{uv}]}{N^2}. \tag{3.1.9}$$

Similarly we can estimate $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})$ by

$$P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|E_{uv} = 1, \Theta_M) \approx \frac{\sum_{(i,j)\in\widetilde{\mathcal{E}}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{|\widetilde{\mathcal{E}}|}, \tag{3.1.10}$$

$$P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|\Theta_M) \approx \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{N^2}. \tag{3.1.11}$$

Note that the estimation of $P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|E_{uv} = 1, \Theta_M)$ is based on $\widetilde{\mathcal{E}}$, which is unfortunately from the implicit bias-free graph $\widetilde{\mathcal{G}}$ and unobservable. But we can approximate it with $\mathcal{E}$ in the following way: after grouping node pairs by non-sensitive attribute value combinations $\widetilde{\boldsymbol{a}}_{uv}$, the sensitive attributes only re-route the edges but do not change the number of edges in each group. Thus,

$$\begin{aligned} P_{\widetilde{o}}(\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}|E_{uv} = 1, \Theta_M) &\approx \frac{\sum_{(i,j)\in\widetilde{\mathcal{E}}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{|\widetilde{\mathcal{E}}|} \\ &= \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V},\widetilde{\boldsymbol{a}}_{ij}=\widetilde{\boldsymbol{a}}_{uv}} \mathbb{I}[(i,j)\in\widetilde{\mathcal{E}}]}{|\widetilde{\mathcal{E}}|} \\ &= \frac{\sum_{i\in\mathcal{V},j\in\mathcal{V},\widetilde{\boldsymbol{a}}_{ij}=\widetilde{\boldsymbol{a}}_{uv}} \mathbb{I}[(i,j)\in\mathcal{E}]}{|\widetilde{\mathcal{E}}|} \\ &= \frac{\sum_{(i,j)\in\mathcal{E}} \mathbb{I}[\widetilde{\boldsymbol{a}}_{ij} = \widetilde{\boldsymbol{a}}_{uv}]}{|\mathcal{E}|}. \end{aligned} \tag{3.1.12}$$

For node pairs with the same attribute value combination, Eq (3.1.8)-Eq (3.1.11) only need to be calculated once instead of for each pair. This can be done by first grouping node pairs by their attribute value combinations and then perform estimation in each group. However, when there are many attributes or attributes can take many unique values, the estimates may become inaccurate since there will be many groups and each group may only have a few nodes. In this case, we can make independence assumptions among the attributes. For example, by assuming they are independent, the estimate for a specific attribute value combination over all the K attributes becomes the product of K estimates, one for each attribute. The non-sensitive attributes can be safely removed under this assumption with $\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv}) = 1$, and only $R(\boldsymbol{a}_{uv})$ needs to be estimated as $R(\boldsymbol{a}_{uv}) = \prod_{i=1}^{m} R(\boldsymbol{a}_{uv}[i])$. Since UGE-W only assigns pre-computed weights to the loss, the optimization based on it will not increase the complexity of any graph embedding method.

Regularization-Based UGE

We propose an alternative way for UGE which adds a regularization term to the loss function that pushes the embeddings to satisfy properties required by the bias-free graph $\widetilde{\mathcal{G}}$. Specifically, when the node embeddings are learned from $\widetilde{\mathcal{G}}$, their produced edge distributions should be the same with and without the sensitive attributes. To enforce this condition, we need to regularize the discrepancy between $P_o(E_{uv} = 1|C = \boldsymbol{a}_{uv}, \Theta_M)$ and $P_{\widetilde{o}}(E_{uv} = 1|\widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M)$ induced from the node embeddings. We can use the scores in $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ as a proxy to represent edge probability produced by the embeddings of nodes u and v, i.e., high $\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)$ indicates high probability of an edge between u and v. We can measure $P_o(E_{uv} = 1|C = \boldsymbol{a}_{uv}, \Theta_M)$ by aggregating node pairs with the same attribute value combination to marginalize out the effect of Θ_M and

Table 3.1: Statistics of evaluation graph datasets.

Statistics	**Pokec-z**	**Pokec-n**	**MovieLens-1M**
# of nodes	$67,796$	$66,569$	$9,992$
# of edges	$882,765$	$729,129$	$1,000,209$
Density	0.00019	0.00016	0.01002

focus on the influence from attributes as

$$Q_{\boldsymbol{a}_{uv}} = \frac{1}{N_{\boldsymbol{a}_{uv}}} \sum_{i \in \mathcal{V}, j \in \mathcal{V}, \boldsymbol{a}_{ij} = \boldsymbol{a}_{uv}} \mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_i, \boldsymbol{z}_j), \tag{3.1.13}$$

where we use $Q_{\boldsymbol{a}_{uv}}$ to denote the approximated measure of $P_o(E_{uv} = 1 | C = \boldsymbol{a}_{uv}, \Theta_M)$, and $N_{\boldsymbol{a}_{uv}}$ is the number of node pairs that has the attribute value combination $\boldsymbol{a}_{uv}$. For pairs with the same attribute value combination, $Q_{\boldsymbol{a}_{uv}}$ only needs to be calculated once. Similarly, $P_{\widetilde{o}}(E_{uv} = 1 | \widetilde{C} = \widetilde{\boldsymbol{a}}_{uv}, \Theta_M)$ can be represented by $Q_{\widetilde{\boldsymbol{a}}_{uv}}$, which can be obtained by aggregating the scores over pairs with non-sensitive attribute value combination $\widetilde{\boldsymbol{a}}_{uv}$. Finally, we use ℓ_2 distance between $Q_{\boldsymbol{a}_{uv}}$ and $Q_{\widetilde{\boldsymbol{a}}_{uv}}$ as the regularization

$$\begin{aligned} &\mathcal{L}_{UGE-R}(\mathcal{G}) \\ &= \sum_{(u,v) \in \mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) + \lambda \sum_{u \in \mathcal{V}, v \in \mathcal{V}} ||Q_{\boldsymbol{a}_{uv}} - Q_{\widetilde{\boldsymbol{a}}_{uv}}||_2, \end{aligned} \tag{3.1.14}$$

where λ controls the trade-off between the per-edge losses and the regularization.

In contrast to adversarial regularizations employed in prior work [149, 152–154], UGE-R takes a different perspective in regularizing the discrepancy between graphs with and without sensitive attributes induced from the embeddings. All previous regularization-based methods impose the constraint on individual edges. We should note that the regularization term is summed over all node pairs, which has a complexity of $O(N^3)$ and can be costly to calculate. But in practice, we can add the regulariztaion by only sampling batches of node pairs in each iteration during model update, and use λ to compensate the strength of the regularization.

Combined Method

As hinted, UGE-W is a sufficient condition for unbiased graph embedding, since it directly learns node embeddings from a bias-free graph. UGE-R is a necessary condition, as it requires the learned embeddings to satisfy the properties of a bias-free graph. We can combine them to trade-off the debiasing effect and utility,

$$\begin{aligned} &\mathcal{L}_{UGE-C}(\mathcal{G}) \\ &= \sum_{(u,v) \in \mathcal{E}} \mathcal{L}_{edge}(\mathrm{s}_{\boldsymbol{\theta}}(\boldsymbol{z}_u, \boldsymbol{z}_v)) \frac{\widetilde{R}(\widetilde{\boldsymbol{a}}_{uv})}{R(\boldsymbol{a}_{uv})} + \lambda \sum_{u \in \mathcal{V}, v \in \mathcal{V}} ||Q_{\boldsymbol{a}_{uv}} - Q_{\widetilde{\boldsymbol{a}}_{uv}}||_2, \end{aligned} \tag{3.1.15}$$

where we use $\mathcal{L}_{UGE-C}(\mathcal{G})$ to represent the combined method. $\mathcal{L}_{UGE-C}(\mathcal{G})$ thus can leverage the advantages of both UGE-W and UGE-R to achieve better trade-offs between the unbiasedness and the utility of node embeddings in downstream tasks.

3.1.5 Experiments

In this section, we study the empirical performance of UGE on three benchmark datasets in comparison to several baselines. In particular, we apply UGE to *five* popularly adopted backbone graph embedding models to show its wide applicability. To evaluate the debiasing performance, the node embeddings are firstly evaluated by their ability to predict the value of sensitive attributes, where lower prediction performance means better debiasing effect. Then a task-specific metric is used to evaluate the utility of the embeddings. Besides, we also apply fairness metrics in the link prediction results to demonstrate the potential of using embeddings from UGE to achieve fairness in downstream tasks.

Table 3.2: Unbiasedness evaluated by Micro-F1 on Pokec-z and Pokec-n. Bold numbers highlight the best in each row.

Dataset	Model	Prediction Target	No Debiasing	Fairwalk	CFC	UGE-W	UGE-R	UGE-C	Random
Pokec-z	GAT	Gender (Micro-F1)	0.6232	0.6135	0.5840	0.6150	0.6094	**0.5747**	0.4921
		Region (Micro-F1)	0.8197	0.8080	0.7217	0.6784	0.7660	**0.6356**	0.4966
		Age (Micro-F1)	0.0526	0.0522	0.0498	0.0431	0.0545	**0.0429**	0.0007
Pokec-n	node2vec	Gender (Micro-F1)	0.5241	0.5291	0.5241	0.5187	**0.5095**	0.5158	0.5078
		Region (Micro-F1)	0.8690	0.8526	0.8423	0.8158	0.6975	**0.6347**	0.4987
		Age (Micro-F1)	0.0626	0.0534	0.0426	0.0305	0.0294	**0.0194**	0.0002

Setup

Dataset. We use three public user-related graph datasets, Pokec-z, Pokec-n and MovieLens-1M, where the users are associated with sensitive attributes to be debiased. The statistics of these three datasets are summarized in Table 3.1. Pokec[3] is an online social network in Slovakia, which contains anonymized data of millions of users [174]. Based on the provinces where users belong to, we used two sampled datasets named as **Pokec-z** and **Pokec-n** adopted from [165], which consist of users belonging to two major regions of the corresponding provinces, respectively. In both datasets, each user has a rich set of features, such as education, working field, interest, etc.; and we include *gender, region* and *age* as (sensitive) attributes whose effect will be studied in our evaluation. **MovieLens-1M**[4] is a popular movie recommendation benchmark, which contains around one million user ratings on movies [175]. In our experiment, we construct a bipartite graph which consists of user and movie nodes and rating relations as edges. The dataset includes *gender, occupation* and *age* information about users, which we treat as sensitive attributes to be studied. We do not consider movie attributes, and thus when applying UGE, only user attributes are counted for our debiasing purpose.

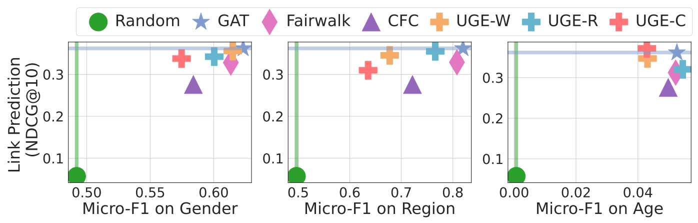
Graph embedding models. UGE is a general recipe for learning unbiased node embeddings, and can be applied to different graph embedding models. We evaluate its effectiveness on five representative embedding models in the supervised setting with the link prediction task. **GCN** [176], **GAT** [177], **SGC** [158] and **node2vec** [59] are deep learning models, and we use dot product between two node embeddings to predict edge probability between them and apply cross-entropy loss for training. **MF** [178] applies matrix factorization to the adjacency matrix. Each node is represented by an embedding vector learned with pairwise logistic loss [144].

Baselines. We consider three baselines for generating unbiased node embeddings. (1) **Fairwalk** [17] is based on node2vec, which modifies the pre-processing of random-walk generation by grouping neighboring nodes with their values of the sensitive attributes. Instead of randomly jumping to a neighbor node, Fairwalk firstly jumps to a group and then sample a node from that group for generating random walks. We extend it to GCN, GAT and SGC by sampling random walks of size 1 to construct the corresponding per-edge losses for these embedding models. (2) **Compositional Fairness Constraints (CFC)** [149] is an algorithmic method, which adds an adversarial regularizer to the loss by jointly training a composition of sensitive attribute discriminators. We apply CFC to all graph embedding models and tune the weight on the regularizer, where larger weights are expected to result in embeddings with less bias but lower utility. (3) **Random** embeddings are considered as a bias-free baseline. We generate random embeddings by uniformly sampling the value of each embedding dimension from [0, 1].

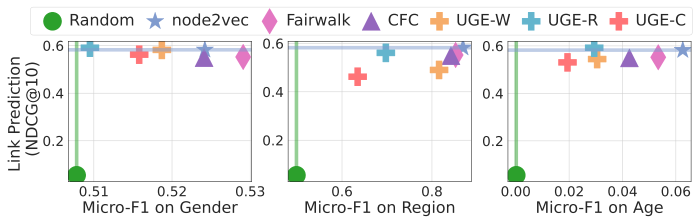
Configurations. For the Pokec-z and Pokec-n datasets, we apply GCN, GAT, SGC and node2vec as embedding models and apply debiasing methods on top of them. For each dataset, we construct positive examples for each node by collecting N_{pos} neighboring nodes with N_{pos} equal to its node degree, and randomly sample $N_{neg} = 20 \times N_{pos}$ unconnected nodes as negative examples. For each node, we use 90% positive and negative examples for training and reserve the rest 10% for testing. For Movielens-1M, we follow common practices and use MF as the embedding model [17, 149]. We do not evalaute Fairwalk on this dataset since there is no user-user connections and fair random walk cannot be directly applied. The rating matrix is binarized to create a bipartite user-movie graph for MF. We use 80% ratings for training and 20% for testing. For all datasets and embedding models, we set the node embedding size to $d = 16$. We include more details about model implementations and hyper-parameter tuning in Appendix A.

[3]https://snap.stanford.edu/data/soc-pokec.html

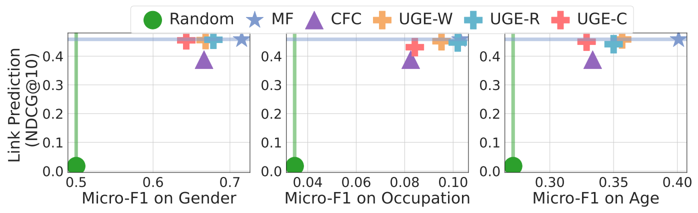
[4]https://grouplens.org/datasets/movielens/1m/



(a) Pokec-z with GAT as embedding model



(b) Pokec-n with node2vec as embedding model



(c) MovieLens-1M with MF as embedding model

Figure 3.2: Trade-off between the utility (by NDCG@10) and unbiasedness (by Micro-F1) of different methods. Random embeddings give the lowest Micro-F1 (green line), and no debiasing gives the best NDCG@10 (blue line). An ideal debiasing method should locate itself at the upper left corner.

In Section 3.1.5, we compare the unbiasedness and utility of embeddings from different baselines. We evaluate fairness resulted from the embeddings in Section 3.1.5. Since there is a large number of experimental settings composed of different datasets, embedding models, and baselines, we report results from different combinations in each section to maximize the coverage in each component, and include the other results in Appendix B.

Unbiasedness and Utility Trade-off

We firstly compare the unbiasedness of node embeddings from different debiasing methods. For each sensitive attribute, we train a logistic classifier with 80% of the nodes using their embeddings as features and attribute values as labels. We then use the classifier to predict the attribute values on the rest of 20% nodes and evaluate the performance with Micro-F1. The Micro-F1 score can be used to measure the severity of bias in the embeddings, i.e., a lower score means lower bias in the embeddings. Random embeddings are expected to have the lowest Micro-F1 and embeddings without debiasing should have the highest Micro-F1. We show the results on Pokec-z with GAT as base embedding model and Pokec-n with node2vec as the base embedding model in Table 3.2. From the results, we see that embeddings from UGE methods always have the least bias against all baselines with respect to all sensitive attributes and datasets. This confirms the validity of learning unbiased embeddings from a bias-free graph. Besides, by combining UGE-W and UGE-R, UGE-C usually produces the best debiasing effect, which demonstrates the complementary effect of the two methods.

Besides the unbiasedness, the learned embeddings need to be effective when applied to downstream tasks. In particular, we use NDCG@10 evaluated on the link prediction task to measure the utility of the embeddings. Specifically, for each target node, we create a candidate list of 100 nodes that includes all its observed neighbor nodes in the test set and randomly sampled negative nodes. Then NDCG@10 is evaluated on this list with predicted edge probabilities from the node embeddings. Figures 3.2a and 3.2b show the unbiasedness as well as the utility of embeddings from different methods in correspondence to the two datasets and embedding models in Table 3.2. Figure 3.2c shows the results on MovieLens-1M with MF as the embedding model.

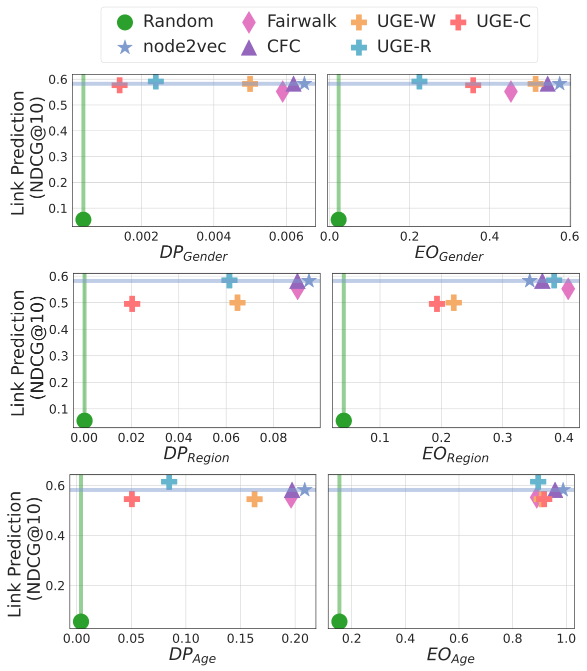


Figure 3.3: Fairness metrics evaluated on link prediction task on Pokec-n with node2vec as the embedding model.

In these plots, different embedding methods are represented by different shapes in the figures, and we use different colors to differentiate UGE-W, UGE-R and UGE-C. Random embeddings do not have any bias and provide the lowest Micro-F1 (green line), while embeddings without any debiasing gives the highest NDCG@10 (blue line). To achieve the best utility-unbiasedness trade-off, an ideal debiasing method should locate itself at the upper left corner. As shown in the figures, UGE based methods achieve the most encouraging trade-offs on these two contradicting objectives in most cases. UGE-C can usually achieve better debiasing effect, without sacrificing too much utility. UGE-W and UGE-R maintain high utility but are less effective than the combined version. CFC can achieve descent unbiasedness in embeddings, but the utility is seriously compromised (such as in Pokec-z and MovieLens-1M). Fairwalk unfortunately does not present an obvious debiasing effect.

High-Level Fairness from Embeddings

We study whether the debiased embeddings can lead to fairness in downstream tasks. We adopt two popular metrics—*demographic parity* (DP) and *equalized opportunity* (EO) to evaluate the fairness of link prediction results from the embeddings. DP requires that the predictions are independent from sensitive attributes, measured by the maximum difference of prediction rates between different combinations of sensitive attribute values. EO measures the independence between true positive rate (TPR) of predicted edges and sensitive attributes. It is defined by the maximum difference of TPRs between different sensitive attribute value combinations. For both DP and EO, lower values suggest better fairness. We use the exact formulation of DP and EO in [155] and use the sigmoid function to convert the edge score for a pair of nodes to a probability.

We show the results on fairness vs., utility in Figure 3.3, which are evaluated on each of the three sensitive attributes in Pokec-n with node2vec as the embedding model. In each plot, x-axis is the DP or EO and y-axis is the NDCG@10 on link prediction. Similar to Figure 3.2, the ideal debiasing methods should locate at the upper left corner. Except for EO on the *age* attribute where all methods performs similarly, UGE methods can achieve significantly better fairness than the baselines on both DP and EO, while maintaining competitive performance on link prediction. UGE-C can achieve the most fair predictions. This study shows UGE's ability to achieve fairness in downstream tasks by effectively eliminating bias in the learned node embeddings.

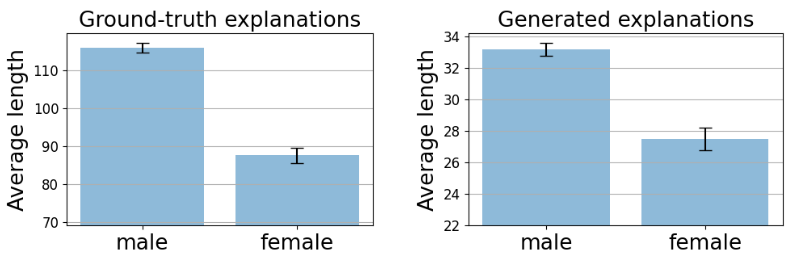


Figure 3.4: Left: average length of ground-truth explanations written by male and female users in the train set of Amazon Games. Right: average length of explanations generated on the test set by PETER.

3.2 Counterfactual Fairness in Personalized Explanation Generation

Enforcing independence between personalized results and sensitive attributes of users/items can be a strong constraint, which, although effective in enforcing fairness, may adversely affect the performance of personalization. Additionally, fairness needs may vary depending on specific aspects of the results, and certain kinds of dependence between the results and the sensitive attributes can be important for personalization. In this case, appropriate fairness measures should be thoughtfully designed to reflect these specific needs without affecting the other aspects of the results. Such measures then serve as objectives for promoting fairness. In this section, we focus on fairness in natural language explanation generation and explores fairness considerations from both individual and group perspectives.

Personalized explanation generation in PS is essentially a personalized text generation (PTG) task [38, 39, 65, 108], which aims to provide a user with a paragraph of text to describe a recommended item according to her preferences. As discussed in Chapter 2, the generators are language models usually trained on user written reviews in online e-commerce platforms [38, 67, 122], where sentences related to item descriptions are retained to construct the ground-truth explanations. However, due to historical, social, or behavioral reasons, there may exist bias that associates certain linguistic characteristics of the review text with the users' protected attributes, e.g., gender, race, education, etc. [43, 179, 180]. Some of the characteristics that feature the diversity of language use [179, 181] are proper to be captured by personalization. While others that are about the *linguistic quality* of the reviews [182], such as informativeness, fluency, structure, etc., should not be kept in explanation generation, which otherwise will lead to unfair treatments when serving users.

As an example, we study the explanation generation on Amazon Games[5] with the personalized transformer (PETER) model [39]. In this case study, we use length as a proxy of the text quality, since length has been shown to strongly correlate with different linguistic quality traits as well as human-judged text quality [182–185]. As shown in Figure 3.4 (left), we find that male users generally write longer and more detailed reviews about games than female users. If directly trained on such data, a language model can inherit the bias and generate explanations discriminately towards users' gender—when a new game is recommended, the model generates short explanations for female users, but long and detailed explanations for male users, as indicated in Figure 3.4 (right).

We take a causal perspective to analyze the bias and enforce counterfactual fairness (CF) [51] in personalized explanation generation: the quality of the generated explanations should not differentiate a user in the real world vs. in the counterfactual world where the user's protected attribute is changed (e.g., male $\rightarrow$ female). Although CF has been studied in other NLP tasks [186, 187], they mainly study how the model's outputs depend on the attribute-specific words directly encapsulated in the input text [48–50]. So counterfactual inference (CI) can be easily performed on the text itself, e.g., changing male pronouns to female pronouns [186, 187]. How the quality of generated text can be unfair towards the protected attributes of users being served has not been studied before. In particular, user and item IDs are usually input to the model for personalized explanation generation, which are represented by embedding vectors [37, 39, 108]. The user's protected attribute is implicitly encoded in the learned representations [188], and cannot be directly manipulated for CI.

[5]The details about experiment setup is in Section 3.2.5.

To achieve the goal, we develop a general framework, COFFEE, for **CO**unter**F**actual **F**airn**E**ss in **E**xplanation generation. In COFFEE, we treat a user's protected attribute value as a separate token input to the model, and disentangle its representation from the user's representation [189–191]. By switching between real and counterfactual attribute values for CI, we impose a measure-specific CF constraint [54] on the quality of generated explanations. We then develop a novel fair policy learning scheme to optimize CF. We use user-side fairness by default in discussions, but as we will show, COFFEE is general to ensure fairness on either user or item side in the two-sided market [31], and be adapted to different models. Extensive experiments show COFFEE's superiority in comparison to baselines.

3.2.1 Related Work

Uniqueness and Importance: Fairness in machine learning (ML) is originally studied in automatic decision-making systems which directly impose a "significant" or "legal" effect on individuals [45]. The fairness considerations are mostly on *resource allocation* by ML models, such as loan assessment [46], job application [47], where the model predictions can unfavorably affect a protected group [192, 193].

Instead of studying fairness of resource allocation, NLP researchers focus on the *representational fairness* [48–50] of how language models shape social biases and stereotypes through natural language understanding (NLU) or generation (NLG). For example, bias in natural language understanding tasks, In NLU, as an example, Kiritchenko et al. [194] find that sentiment analysis systems yield different sentiment scores for sentences containing names associated with African Americans vs. European Americans. The fairness in NLG cares how generated text may contain biased information about a specific demographic group. Huang et al. [186] analyze the sentence completion by a GPT-2 model, and find different sentiment distributions of the completed sentences when the occupation word is changed in the prompts. Bordia et al. [44] show that *doctor* co-occurs more frequently with male pronouns while *nurse* co-occurs more often with female pronouns in generated text. However, these biases are easier to analyze, since both the demographic and the bias signals are encapsulated within the text. To the best of our knowledge, we are the first to study how personalization links the bias in NLG to protected attributes of users being served.

Fairness Notions: There exist different fairness notions in the literature, among which group-wise fairness notions are the firstly studied ones [23–25]. However, group-wise fairness has different quantitative definitions and different criteria such as demographic parity or equal opportunity are incompatible in general [195, 196]. Depending on the relationship between the protected attribute and the data, some definitions can even increase discrimination [51]. Individual fairness [52, 53] requires similar users to receive similar predictions. But it depends on the specific distance metric for similarity, which must be carefully chosen, and requires an understanding of the domain at hand [197]. In contrast, counterfactual fairness (CF) [51] considers fairness from a causal perspective and has been widely adopted recently as a more robust fairness notion [54–56], which also improves group-wise fairness in certain scenarios [198, 199].

3.2.2 Problem Formulation

In the following discussions, we consider a single protected attribute on the user side. But the proposed framework can be easily generalized to multiple attributes on either the user or the item side. The value of a user's protected attribute is denoted by a variable $A \in \mathcal{A}$, where $\mathcal{A}$ is the set of possible attribute values, e.g., $\mathcal{A} = \{male,\ female\}$ for gender. Each entry of the dataset is a tuple of (u, i, a, s, e), corresponding to user ID, item ID, observed protected attribute value, rating, and ground-truth explanation. The explanation generator G_θ is a language model parameterized by θ. Given a user u, an item i, and the observed attribute value a, an explanation can be sampled from the generator as $Y \sim G_\theta(u, i|A = a)$. The linguistic quality of any sampled explanation y is measured by a function $Q(y)$. In particular, we assume Q is a *black box oracle* and can only be queried, which is specified according to the fairness needs. This design is essential for practical uses, since the linguistic quality can be specified differently by the designer in different applications and be evaluated in different ways, such as via an explicit function, human-evaluation, or a government provided tool. Without loss of generality, we assume higher Q means higher quality. Counterfactual fairness (CF) on the linguistic quality of generated explanations requires that, given a user u and an item i,

$$P(Q(Y_{A \leftarrow a})|u, i, a) = P(Q(Y_{A \leftarrow a'})|u, i, a) \tag{3.2.1}$$

where $Y_{A\leftarrow a'} \sim G_\theta(u, i|A = a')$ is the explanation generated when we counterfactually assign the value of the user's protected attribute by $A \leftarrow a', a' \neq a$. The right side of eq. (3.2.1) evaluates the quality distribution of explanations generated *had the user's protected attribute value been* a', given that the *observed attribute value is* a [51, 200].

Denote the loss of the generator for a given user u and item i by $\mathcal{L}_{gen}(G_\theta(u, i|A = a), e)$, which is usually the negative log-likelihood (NLL) loss or a combination of several losses [37–39]. We consider training the generator for fair explanation generation as a constrained optimization problem:

$$\begin{aligned} &\min \mathcal{L}_{gen}(G_\theta(u, i|A = a), e) \\ &s.t.\ \mathbb{E}_{Y_{A\leftarrow a}}[Q(Y_{A\leftarrow a})|u, i, a] = \\ &\quad \mathbb{E}_{Y_{A\leftarrow a'}}[Q(Y_{A\leftarrow a'})|u, i, a] \end{aligned} \tag{3.2.2}$$

For ease of presentation, we consider a single user-item pair, and the total loss on a dataset is simply summed over all user-item pairs with the constraint applied to every pair. In this work, we apply the first-order moment of the quality of generated explanations to construct the constraint, and leave the extension to other moment-matching constraints for future work. We further simplify the expression of the constraint as $\mathbb{E}[Q(Y_{A\leftarrow a})] = \mathbb{E}[Q(Y_{A\leftarrow a'})]$.

3.2.3 COFFEE

Disentangled Attribute Representation

As discussed, the protected attribute and the user's preferences are entangled in a user's history, and the attribute information is implicitly encoded in the user's representation, making it impossible to perform CI. To overcome this, we consider a user's protected attribute value as a separate token input to the model along with the user and item IDs, and propose to learn disentangled attribute representations [188–191]. This is similar to works in controllable text generation [201–203], where disentangled attribute tokens are used as input to control the topic, sentiment, or style of the generated text.

For a given tuple (u, i, a), we denote the representation for the attribute value a as $\mathbf{r}_a$, the user's preference representation (independent from the protected attribute) as $\mathbf{r}_u$ and item representation as $\mathbf{r}_i$. The complete user representation is $\mathbf{r}_u^a = [\mathbf{r}_a, \mathbf{r}_u]$. Correspondingly, when performing $A \leftarrow a'$ on user u, we change the input attribute token from a to a', and the new user representation becomes $\mathbf{r}_u^{A\leftarrow a'} = [\mathbf{r}'_a, \mathbf{r}_u]$. Note that each attribute value has its own representation, and is shared across all users having that same attribute value. For instance, all male users' attribute representation is the same vector $\mathbf{r}_{male}$. We can do the same for item-side attributes as $\mathbf{r}_i^a = [\mathbf{r}_a, \mathbf{r}_i]$.[6]

Simply separating the user's protected attribute and preference representations can not guarantee that $\mathbf{r}_u$ will not contain any information about the protected attribute, inhibiting the accuracy of CI. To further enforce the disentanglement, we introduce a discriminator $D(\mathbf{r}_u)$, and add an adversarial loss on $\mathbf{r}_u$ in eq. (3.2.2) as

$$\begin{aligned} \min\ & \mathcal{L}_{gen}(G_\theta(u, i|A = a), e) + \lambda_D \log(D(r_u, a)) \\ s.t.\ & \mathbb{E}[Q(Y_{A\leftarrow a})] = \mathbb{E}[Q(Y_{A\leftarrow a'})] \end{aligned} \tag{3.2.3}$$

where $D(r_u, a)$ is the probability of predicting the correct attribute value a by the discriminator. In this way, we adversarially remove the protected attribute information from $\mathbf{r}_u$, and enforce $\mathbf{r}_a$ to capture all the attribute information. During mini-batch optimization, we alternate between the parameter updates of the model and the discriminator as follows: (1) X batches of updates minimizing the loss eq. (3.2.3) with D fixed, and (2) Z batches of updates maximizing the the loss eq. (3.2.3) with the generator G_θ fixed.

Policy Learning for Fairness

Once the user, item, and attribute representations are learned and *fixed*, we can perform and optimize the generator w.r.t. the CF constraint. Due to the non-convexity of the constrained fairness optimization problem,

[6] To generalize to K protected attributes, we can simply add K attribute tokens in the input, each of which is mapped to its separately learned representations.

closed-form solutions are impossible. We add the constraint with Lagrangian relaxation as a regularization in the loss [54]

$$\min \mathcal{L}_{all} + \lambda \big| \mathbb{E}[Q(Y_{A\leftarrow a})] - \mathbb{E}[Q(Y_{A\leftarrow a'})] \big| \tag{3.2.4}$$

with $\mathcal{L}_{all} = \mathcal{L}_{gen}(G_\theta(u, i|A = a), e) + \lambda_D \log(D(r_u, a))$ in eq. (3.2.3) denotes all other losses except for the CF constraint, and λ is a hyper-parameter for fairness-utility trade-off.

However, standard gradient methods can not be directly applied to optimize this constraint—the explanations are discretely sampled from the generator, and the quality function Q is an oracle. Instead, we consider policy learning for fairness optimization, where the explanation distribution imposed by the generator is considered as the policy for explanation generation, and the sampled explanations are actions. We then derive the policy gradients on the generator's parameters based on carefully designed rewards of sampled explanations. Concretely, for estimating the expectations in the regularization, we sample N explanations and calculate the average quality for explanations sampled in both the real-world and the counterfactual world. Denote the regularization term as $\mathcal{L}_{fair}$, the expectations in the regularization can be estimated as:

$$\begin{aligned} \mathcal{L}_{fair} &= \big| \mathbb{E}[Q(Y_{A\leftarrow a})] - \mathbb{E}[Q(Y_{A\leftarrow a'})] \big| \\ &\approx \operatorname{sign}(\Delta) \left(\frac{1}{N} \sum_{k=1}^{N} Q\big(y^k_{A\leftarrow a}\big) - \frac{1}{N} \sum_{k=1}^{N} Q\big(y^k_{A\leftarrow a'}\big) \right) \end{aligned} \tag{3.2.5}$$

$\Delta = \frac{1}{N} \sum_{k=1}^{N} Q\big(y^k_{A\leftarrow a}\big) - \frac{1}{N} \sum_{k=1}^{N} Q\big(y^k_{A\leftarrow a'}\big)$. For an explanation $y^k_{A\leftarrow a}$ sampled in the real-world, its contribution to the unfairness in regularization term is thus $\frac{1}{N} \operatorname{sign}(\Delta) Q\big(y^k_{A\leftarrow a}\big)$. Since we are minimizing the regularization to improve fairness, the reward for $y^k_{A\leftarrow a}$ is considered as $r\big(y^k_{A\leftarrow a}\big) = -\frac{1}{N} \operatorname{sign}(\Delta) Q\big(y^k_{A\leftarrow a}\big)$. Similarly, the reward for the a sampled explanation $y^k_{A\leftarrow a'}$ in the counterfactual world is $r\big(y^k_{A\leftarrow a'}\big) = \frac{1}{N} \operatorname{sign}(\Delta) Q\big(y^k_{A\leftarrow a'}\big)$. In this way, we convert the CF optimization into maximizing the rewards of generated explanations.

Although the rewards are designed to minimize the difference in the qualities of explanations generated in the real vs. counterfactual world, we have no control over how the difference should be minimized during optimization. The model may arbitrarily improve or decrease the quality of explanations to achieve fairness, e.g., always generating low quality explanations, which greatly hurt their utility in practice. To address this issue, we further design a weighting mechanism to calibrate the rewards such that the optimization can focus more on improving (or decreasing) the quality measure for achieving fairness, which will empower the designer to better control the fairness optimization and the utility-fairness trade-off. Specifically, we introduce a quality promotion weight $\eta \in [0, 1]$ to re-weigh the rewards of explanations in the world with lower expected quality, and use $1 - \eta$ to reweigh the rewards of explanations in the other world. The resulting calibrated rewards are:

$$\begin{aligned} r_w\big(y^k_{A\leftarrow a}\big) &= -\frac{1}{N} \operatorname{sign}(\Delta) Q\big(y^k_{A\leftarrow a}\big) \cdot w(\Delta) \\ r_w\big(y^k_{A\leftarrow a'}\big) &= \frac{1}{N} \operatorname{sign}(\Delta) Q\big(y^k_{A\leftarrow a'}\big) \cdot (1 - w(\Delta)) \end{aligned}$$

where $w = \frac{\operatorname{sign}(\Delta)+1}{2}(1-\eta) + \left(1 - \frac{\operatorname{sign}(\Delta)+1}{2}\right)\eta$. We can leverage this weight to guide the fairness optimization—if $\eta > 0.5$, the algorithm will focus more on improving the quality of low-quality explanations for fairness. Finally, for stability and faster convergence of training, we apply the advantage trick [204] to regularize the rewards, and each reward becomes its difference from the average reward in its corresponding world:

$$\begin{aligned} r_{adv}\big(y^k_{A\leftarrow a}\big) &= r_w\big(y^k_{A\leftarrow a}\big) - \bar{r}_w\big(y^k_{A\leftarrow a}\big) \\ r_{adv}\big(y^k_{A\leftarrow a'}\big) &= r_w\big(y^k_{A\leftarrow a'}\big) - \bar{r}_w\big(y^k_{A\leftarrow a'}\big) \end{aligned} \tag{3.2.6}$$

The policy gradient [38, 205] on θ can then be estimated as:

$$\begin{aligned} \nabla_\theta \mathcal{L}_{fair} \approx &\sum_{k=1}^{N} \nabla_\theta \log G_\theta\big(y^k_{A\leftarrow a}\big) r_{adv}\big(y^k_{A\leftarrow a}\big) \\ &+ \sum_{k=1}^{N} \nabla_\theta \log G_\theta\big(y^k_{A\leftarrow a'}\big) r_{adv}\big(y^k_{A\leftarrow a'}\big) \end{aligned} \tag{3.2.7}$$

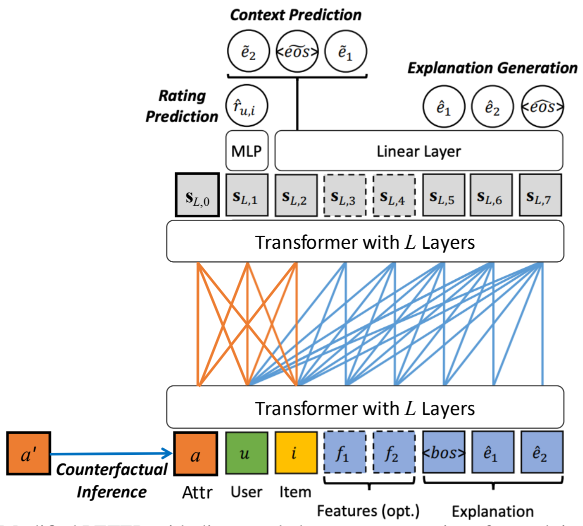


Figure 3.5: Modified PETER with disentangled user representations for applying COFFEEE.

Intuitively, during optimization, the probability of sampled explanations that lead to the unfairness will be demoted, while the probability of those that contribute to fairness will be promoted relatively.

To train a COFFEE model, we first pre-train the model without the fairness constraint. Then we fix the latent representations of users, items and attributes, and add the fairness constraint in the objective function for fine-tuning the model.

3.2.4 Applying COFFEE to Existing Models

Existing models for personalized explanation generation mostly use either Transformer [39] or RNN [37,38,65] as the generator. In this section, we demonstrate the application of COFFEE to PETER [39], which is based on Transformer and is the state-of-the-art explanation generation model. In Appendix C, we also illustrate the application of COFFEE to NRT [37], which is based on RNN.

In PETER [39], the user and item IDs are treated as two special tokens appended at the beginning of the explanation word sequence before inputting to transformer layers. The user and the item tokens can attend to each other, while the word tokens can only attend to past tokens. Either the user or the item embedding can capture the information of this user-item pair. For better personalization, a context prediction loss is introduced in addition to the NLL loss, which aims to predict the words in the explanation without caring about their order. Besides, rating prediction is made possible by feeding the output user or item embedding to an MLP. We denote the overall loss as $\mathcal{L}_{peter}$, which is the summation of the NLL loss, context prediction loss, and MSE for rating prediction (Section 4 in [39]).

To apply COFFEE to PETER, we first add an attribute token at the beginning of each sequence for disentangled attribute embedding, as shown in Figure 3.5. The discriminator for better disentanglement is applied on user or item embedding at the input embedding layer. The user, item and attribute tokens can attend to each other, while the explanation words can only attend to past tokens. The loss of applying COFFEE to PETER is constructed by replacing $\mathcal{L}_{gen}$ in eq. (3.2.3) by $\mathcal{L}_{peter}$.

3.2.5 Experiments

We conduct comprehensive experiments on three public review datasets: Amazon Video Games, Amazon Movies & TV, and Yelp Restaurant, and compare COFFEE with multiple baselines. Due to space limit, we present the results based on PETER, and put examples of generated explanations and the results based on NRT in Appendix D.6.

Table 3.3: Statistics of the datasets and the protected attributes used in experiments. The candidate values of the attributes are indicated in parentheses.

	#Users	#Items	#Reviews	Attribute Values
Video Games	9,511	5,173	59,374	male, female
Movies & TV	28,001	12,888	265,646	male, female
Yelp	35,714	24,900	1,499,291	\$, \$\$, \$\$\$

Experiment Setup

Datasets. In Amazon *Video Games* and the *Movies & TV* review datasets[7], we study use users' binray gender (male, female) as the protected attribute. We also study the fairness on the item side on the Yelp dataset[8], where a restaurant's price range (\$, \$\$, \$\$\$) is used as the protected attribute, as the designer may require the system not to discriminate a restaurant simply because the restaurant sets lower prices for equally good food and service than its counterparts. The statistics of the datasets are summarized in Table 3.3. The details of data processing are described in Appendix D. We split each dataset into training (80%), validation (10%), and testing (10%) sets, and ensure that there is at least one record in each subset for every user and item.

Baselines. As we are the first to study the quality fairness of personalized text generation, there is no existing work that directly addresses this problem. For comparison, we adapt popular methods in the fairness literature for the purpose, including pre/ in/post-processing methods. Below we briefly introduce them and describe their detailed implementations in Appendix D.

- RAW: Original PETER or NRT model.
- ADV (in-processing): Adversarially removing the sensitive information in user or item representations by adding a discriminator [200].
- NORM (pre-processing): Normalizing the training data to remove the bias on group-level.
- BT (pre-processing): Back-translation has been shown to help normalize text and reduce bias [206, 207]. We pre-process the training data by translating the explanations to Chinese and then back to English.
- ATTR: To evaluate the effectiveness of the disentanglement for and optimizing the fairness constraint in COFFEE, we use the model trained without the constraint ($\lambda = 0$) but with disentangled representations as a baseline.
- NATTR (post-processing): We disable the protected attribute token in ATTR for generating explanations during inference.

We train each model on the training set, tune the hyper-parameters on the validation set, and report the results on the testing set. Each result point is averaged over 3 runs. We put the detailed model specifications and parameter tuning in Appendix D.4.

Metrics. Our evaluation consists of two parts: fairness evaluation and utility evaluation.
• Fairness metrics. Fairness requirements are subjective and totally depend on the application and the needs from the system designer [199, 208]. COFFEE is general enough to handle different specifications of fairness. For proof-of-concept and demonstrating the effectiveness of COFFEE, we first use length of the text to specify the quality function as Q_{len}. Besides, one of the main purposes of explanation is to provide informative descriptions on item features for the user to make more informed decisions. We also use the number of item features mentioned in the explanations as a quality measure Q_{feat}. Different quality measures may correlate to each other, and we are interested in seeing how optimizing fairness on one will affect fairness on the other, as well as the compositional effect of optimizing fairness on both [149]. The methods applied to optimize fairness on Q_{len}, Q_{feat}, and $Q_{len} + Q_{feat}$ are denoted as [model]-L/F/LF, respectively. One can simply specify Q to any other quality measures under concern in practice.

We evaluate both individual and group-level fairness, with and without counterfactual perspectives. The detailed formulas of the metrics are in Appendix D. The first metric is Ind-CF, which evaluates the originally defined CF on individuals [51], which is the same as the regularization term in eq. (3.2.4) averaged over all

[7]`https://nijianmo.github.io/amazon/index.html`
[8]https://www.yelp.com/dataset

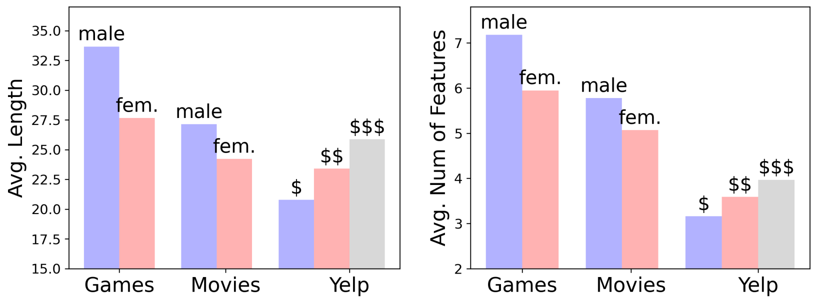


Figure 3.6: The average length (left) and average number of features (right) in generated explanations by PETER on the three datasets.

user-item pairs on the test set. We also evaluate the counterfactual effect on group-level [209]. For each attribute value a and the corresponding demographic group, we counterfactually change the attribute value of all users in this group by $a' \neq a$, and calculate the change on the average quality of this group. We call this metric Grp-CF. Besides counterfactual metrics, we also evaluate COFFEE's generalization to improve group-wise fairness by the popular notion of demographic disparity (DDP) [23]. The CF based metrics can only be evaluated on baselines with disentangled attribute representations for , i.e., ATTR and COFFEE. All fairness metrics are *lower the better*.

• Utility metrics. We adopt several metrics to comprehensively evaluate the utility of generated explanations. Besides the n-gram based BLEU-{1,4} [135] and ROUGE-L [210] scores, we also include the recently prevalent BERTScore [211], which is shown to better capture the semantic meanings between the target and generated text and better correlate to human judgements. We calculate the BERTScore using RoBERTa-large with re-scaled scores. We report F-1 score for ROUGE and BERTScore. Although we focus on explanations, we also evaluate the rating prediction performance by RMSE to examine if different methods will influence the recommendation performance. RMSE is lower the better and the other metrics are higher the better.

Unfairness in Explanation Generation

We first analyze, without any fairness consideration, how Q_{len} and Q_{feat} in model generated explanations can be unfair towards certain protected attributes on users or items, which provides insights into the need of fairness improvement. The statistics of explanations generated on the test sets by PETER is shown in Figure 3.6. On both Amazon Games and Movies & TVs, the model tends to generate longer explanations with more features for male users than female users. On Yelp, the model generates longer and more detailed feature descriptions for more expensive restaurants than cheaper restaurants, potentially users care more about experiences in expensive restaurants as they are paying more. These are consistent with the bias observations in the corresponding training datasets. We also conduct a case study to understand the unfairness from a counterfactual perspective in Appendix D.5. The study proves the effectiveness of our disentangled attribute representation learning, which can also be indicated in the results of ATTR evaluated by Ind-CF and Group-CF in Table 3.4.

Comparison to Baselines

An ideal method should improve fairness without hurting the utility of explanations. We show the complete results on the Amazon Games in Table 3.4 and plot sampled results on the other two datasets in Figure 3.7. The complete results on Amazon Movies & TV and Yelp are in Appendix D.6.

As is shown in Table 3.4, COFFEE achieves superior fairness improvements on all fairness metrics and outperforms baselines with a large margin, which demonstrates its strong ability in promoting fairness. Besides, the comparison between COFFEE and ATTR on Ind-CF and Grp-CF proves the efficacy of optimizing the CF constraint via our fair policy learning scheme. Moreover, optimizing fairness on Q_{len} helps the fairness on Q_{feat}, while the reverse is not as significant. This also indicates that length is a general indicator for text quality. When optimizing on $Q_{len} + Q_{feat}$ together, COFFEE can further improve the fairness on both Q_{len} and Q_{feat}, showing that the correlations between different quality measures can assist the fairness

Table 3.4: Comparison between COFFEE and baselines. BL stand for BLEU and RG denotes ROUGE. BLEU, ROUGE and BERTScore are in percentage values and others are in absolute values. The best results are boldfaced, and the second best are underlined. * indicates $p < 0.05$ for significance test over the second best baseline.

PETER	Fairness on Q_{len}			Fairness on Q_{feat}			Utility						
	Ind-CF↓	Grp-CF↓	DDP↓	Ind-CF↓	Grp-CF↓	DDP↓	BL1↑	BL4↑	RG1↑	RG2↑	RGL↑	BERT↑	RMSE↓
Amazon Games (User's Gender as Protected Attribute)													
RAW	-	-	5.84	-	-	1.18	8.82	1.49	**19.95***	**5.37***	**15.54***	**14.14**	1.21
ADV	-	-	3.04	-	-	0.62	8.78	1.50	16.13	3.96	13.06	12.84	1.21
NORM-L	-	-	5.37	-	-	1.02	8.79	1.54	18.28	4.80	14.60	14.08	1.21
NORM-F	-	-	5.67	-	-	1.13	8.83	1.51	18.20	4.66	14.42	13.91	1.21
BT	-	-	4.00	-	-	0.65	**10.43***	1.55	13.42	2.08	10.44	9.55	1.26
NATTR	-	-	3.24	-	-	0.63	8.76	1.52	16.55	3.86	13.02	12.72	1.21
ATTR	12.42	3.63	6.81	2.68	0.75	1.40	8.91	1.57	17.73	4.45	14.13	14.08	**1.19**
COFFEE-L	3.87	0.47	1.09	1.18	0.03	0.11	10.14	1.24	18.48	4.75	14.69	14.09	**1.19**
COFFEE-F	6.98	2.11	3.99	1.56	0.32	0.68	9.77	**1.58**	16.73	3.98	13.43	14.12	1.20
COFFEE-LF	**3.72***	**0.37***	**0.86***	**1.15**	**0.01**	**0.08***	10.23	1.24	17.38	3.70	13.62	14.02	**1.19**

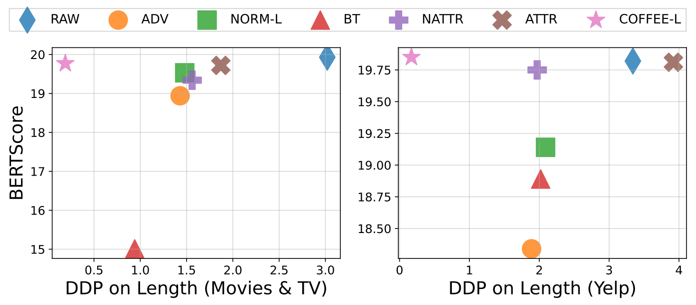


Figure 3.7: Trade-off between fairness and utility on Amazon Movies & TV (left) and Yelp (right) datasets. A method should give low DDP and high BERTScore (top-left corner of each plot) for better trade-offs.

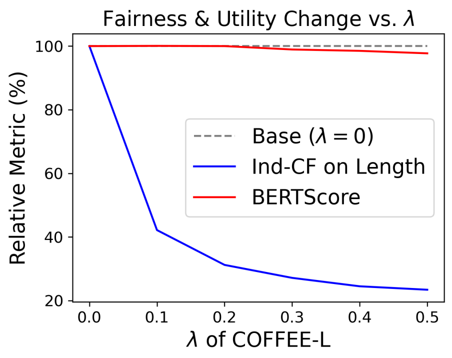


Figure 3.8: Relative Ind-CF and BERTScore ratio w.r.t. to $\lambda = 0$ in COFFEE-L. The Ind-CF and BERTScore of ATTR-PETER are 12.42 and 14.08, respectively.

optimization, similar to the findings in [149]. More importantly, COFFEE achieves strong fairness results while still maintaining high utility on explanation generation.

To better visualize the fairness-utility trade-off, we plot the results on the Amazon Movies & TV and Yelp in Figure 3.7, where we use DDP on Q_{len} for fairness and BERTScore for utility evaluation. COFFEE achieves the best trade-off by drastically reducing DDP with almost no drop on BERTScore. ADV and NORM can also improve fairness at a decent level, but the improvement is less significant than COFFEE, with a similar or even greater decrease on utility. Finally, BT can also improve fairness by simply back-translating the training text from another language, which shows its interesting normalization effect. But BT causes a dramatic sacrifice on utility, as it may eliminate lots of information about personal preference during translation.

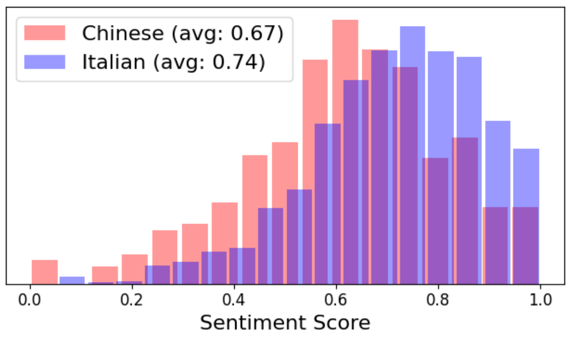


Figure 3.9: The sentiment distributions of PETER generated explanations for Chinese restaurants and Italian restaurants on Yelp. The average sentiment for Chinese restaurant is 0.67, and 0.74 for Italian ones.

Effect of Tuning λ in COFFEE

We further study the fairness-utility trade-off in COFFEE by tuning the weight λ on fairness constraint in eq. (3.2.4). We use Ind-CF on length for fairness evaluation since it corresponds to the constraint that COFFEE directly optimizes, and use BERTScore for utility evaluation. We plot the results on Amazon Games in fig. 3.8, where the y-axis is the metric values ratio to the base results when $\lambda = 0$ (equivalent to ATTR). As λ increases from 0 to 0.5, COFFEE significantly improves the fairness with a drop of about 77% on Ind-CF while barely hurt the BERTScore with a drop on of about 2%. This study shows COFFEE's outstanding efficiency and effectiveness in fairness optimization.

3.3 Group-wise Fairness in Personalized Explanation Generation

COFFEE tries to ensure fairness on each individual from a counterfactual perspective. However, sometimes system designers may care about fairness on a group level, i.e., a certain group should not be disadvantaged by the system. In this section, we further investigate the group-wise fairness in personalized explanation generation.

In this work, we consider the group-wise fairness notion of strong demographic parity, which tries to align the distributions of a specific measure(s) on generated textual explanations for groups defined by certain attributes of users or items. For example, consider the sentiments of generated explanations for Yelp restaurant recommendations by a transformer model PETER [39] in Figure 3.9. After collecting all explanations and evaluate their sentiments, we found that the model tends to generate more positive explanations for certain categories of restaurants (e.g., Italian restaurants), while more negative explanations for others (e.g., Chinese restaurants). Such bias can disadvantage the Chinese restaurants and their business owners, and thus lead to fairness issues.

To quantify such bias, we propose to use the fairness notion of Strong Demographic Parity to investigate fairness, which converts to the fairness metric equivalent to average Wasserstein distance (W-dist) between measure distributions of different groups. Smaller W-dist indicates less bias, and thus better fairness when being used. In particular, the explanation measure does not limit to sentiment, but can also be quality measures, e.g., the diversity of explanations. To achieve such Wasserstein fairness, we propose a general framework that directly minimize W-dist among groups for any specific measure and any sensitive attribute of users or items under interest. Experiments show that our method can significantly improve fairness without hurting the performance of explanation generation.

The closest work to ours is [186], where the authors proposed two methods for achieving Wasserstein fairness. The first method involves adding regularizations on the latent embeddings extracted from the last hidden layers of the language model, which apparently requires access to the model structure and hidden layers. The second method involves adding regularizations of the embeddings from the output layers of a self-trained sentiment classifier, which needs full-control of the classifier used to decide fairness. In comparison, our method improves on several fronts. First, we study fairness with respect to the users or items directly being served by the system-generated text, rather than the bias encapsulated in the text, without analyzing how such bias may affect the end-users of the system. Second, our method generalizes to any measures on generated text, not only sentiments, and we only need the output of any given classifier (e.g., provided by regulators) that evaluates the measure. Third, our method does not require knowledge of the structure or access to parameters of the language model, and only requires fine-tuning based on our fairness objective. Fourth, we directly optimize Wasserstein fairness on generated text.

3.3.1 Framework

Here we use sentiment as the explanation measure to illustrate the proposed framework, but it generalize to any other measures as we will show in experiments. The proposed framework is called WFPTG, meaning Wasserstein-Fair Personalized text Generation. Given a training set of $\mathcal{D} = \{(x_i, a_i, y_i)\}_{\{}i = 1\}^N$, where x_i is a user-item pair as the input for explanation generation, a_i is the sensitive attribute value associated with either the user or the item, and y_i is the observed ratings and reviews. Let the set of possible sensitive attribute values be $\mathcal{A}$, the subset of N_a instances with attribute value a is $\mathcal{D}_a$. The expected sentiment of an instance (x, a) under current policy is $s(x, a)$. The sentiment of a sampled explanation in group $\mathcal{D}_a$ is denoted by a variable S_a and the distribution of sentiment scores of sampled explanations in group $\mathcal{D}_a$ is P_{S_a}. Strong Demographic Parity (SDP) requires that

$$P_{S_a} = P_{S'_a}, \forall a, a' \in \mathcal{A},$$

which is equivalent to

$$\mathbb{E}_{\tau \sim U[0,1]} \left| P\left(S_a > \tau\right) - P\left(S'_a > \tau\right) \right| = 0, \forall a, a' \in \mathcal{A}.$$

Therefore, we can define the measure of violation to SDP by

$$\sum_{a,a' \in \mathcal{A}} \mathbb{E}_{\tau \sim U[0,1]} \left| P\left(S_a > \tau\right) - P\left(S'_a > \tau\right) \right|,$$

which is equivalent to the sum of pairwise Wasserstein-1 distances between any two distributions corresponding to different attribute values. We call this metric Strong Demographic Disparity (SDD) for generated explanations.

Gradient based method for minimizing Wasserstein-1 distance

We propose to achieve SDP by pushing all sentiment distributions to their group-size weighted Wasserstein-1 barycenter distribution $P_{\bar{S}}$, so that to minimize the changes needed for achieving SDP. Note that the sentiment of sampled explanations is a function of language model parameters θ, which is constantly changing during the optimization, and thus the induced barycenter of sentiment distributions are also changing. Therefore, we propose a gradient-based method for optimizing model parameters θ to achieve SDP.

Given a current model parameter θ, we can obtain the empirical sentiment distributions of all groups $\{\widehat{p}_{S_a}\}_{a} \in \mathcal{A}$ based on the sampled explanations in each group, as well as the empirical barycenter $\widehat{P}_{\bar{S}}$. To push each $\widehat{P}_{S_a}$ to $\widehat{P}_{\bar{S}}$, we need to minimize the Wasserstein-1 distance between them, which is calculated as:

$$\mathcal{W}_1\left(\widehat{p}_{S_a}, \widehat{p}_{\bar{S}}\right) = \min_{T_{S_a,\bar{S}} \in \Omega(S_a,\bar{s})} \left\langle T_{S_a,\bar{S}}, C \right\rangle,$$

where

$$\Omega\left(s_a, \bar{s}\right) = \left\{ T \in \mathbb{R}^{N_{s_a}, N_{\bar{s}}} \mid T\mathbf{1}_{\bar{s}} = \frac{1}{N_{s_a}}\mathbf{1}_{s_a}, T^\top \mathbf{1}_{s_a} = \frac{1}{N_{\bar{s}}}\mathbf{1}_{\bar{s}} \right\}.$$

$\mathbf{1}_{s_a}$ denotes a vector of ones of size N_{s_a}, and $\langle \cdot, \cdot \rangle$ is the trace dot product. C is the cost matrix defined by the Wasserstein-1 cost function, with each element $C_{i,j} = \left| s_a^i - \bar{s}^j \right|$. Then the loss to achieve SDP by minimizing the Wasserstein-1 distance between each group and the barycenter is called Strong Demographic Distances (SDD):

$$SDD(\theta) = \sum_{a \in \mathcal{A}} \mathcal{W}_1\left(\widehat{p}_{S_a}, \widehat{p}_{\bar{S}}\right).$$

We can prove that this SDD loss provides a tight bound on the SDD measure defined at the beginning [Jiang et al., UAI'20]. The gradient for the above loss can be derived as

$$\nabla_\theta SDD(\theta) = \sum_{a \in \mathcal{A}} \sum_{i \in [N_{s_a}], j \in [N_{\bar{s}}]} T^*_{s_a,\bar{s}} \nabla_\theta \left| s^i_a - \bar{s}^j \right| .$$

$T^*_{s_a,\bar{s}} = \arg\min_{T_{s_a,\bar{s}} \in \Omega(s_a,\bar{s})} \langle T_{s_a,\bar{s}}, C \rangle$ is the optimal coupling resulting from the Wasserstein-1 distance calculation.

As indicated in the derived gradients, our goal is to move the sentiment of generated explanations closer to the barycenter distribution. However, since the explanation sample procedure is discrete and the sentiment classifier is considered a black-box, it is intractable to directly obtain the gradient $\nabla_\theta \left| s^i_a - \bar{s}^j \right|$. We instead consider policy gradients to achieve the goal. In particular, we can consider the reward of the sentiment for the i-th instance in group with attribute value a as:

$$r\left(s^i_a\right) = -\sum_{j \in [N_{\bar{s}}]} T^*_{s_a,\bar{s}} \left| s^i_a - \bar{s}^j \right| .$$

Thus the policy gradient estimate for maximizing the reward (minimizing the W-1 distance to the barycenter) is derived as $\nabla_\theta \log \pi_\theta \left(x_i, a\right) \cdot r\left(s^i_a\right)$.

Based on the derived policy gradient, we can perform gradient updates for optimizing the SDP. To improve efficiency, we calculate the barycenter every K batches of update. We also maintain a sentiment map, which is updated after each training step only for the instances in the current batch. The full algorithm is in Algorithm 1.

Algorithm 1 Wasserstein-Fair Personalized Text Generation.

```
    Input: policy $\pi_\theta$, training set $\mathcal{D}\{(x_i, a_i, y_i)\}_{i=1}^N$, attribute values $\mathcal{A}$,
           a sentiment map $s(e_{x,a})$ initialized by sampling explanations
           for every instances in the training set (e.g., $s(e_{x,a})$ is estimated
           from sampled explanations $e_{x,a} \sim \pi_\theta(x, a)$), intervals for
           barycenter calculation $K$, sample size from barycenter $\bar{N}$
 1: for $j$-th batch $\mathcal{B}$ in $\mathcal{D}$ do
 2:     if $j\%K == 0$ do
 3:         Calculate the barycenter $\widehat{P}_{\bar{s}}$ and sample $\{\bar{s}_i\}_{i=1}^N$
 4:     Compute the optimal coupling matrices $\left\{T^*_{s_a,\bar{s}}\right\}_{a \in \mathcal{A}}$
 5:     Compute the policy gradient for every instance in $\mathcal{B}$
 6:     Update model parameters based on policy gradients
 7:     Update sentiment map for every instance in $\mathcal{B}$
 8: Output $\pi_\theta$
```

3.3.2 Experiments

We conduct experiments on two public review datasets: Amazon Movies & TV, and Yelp Restaurant, which are the same as used in Section 3.2.5. But for Yelp restaurants, we use the restaurant categories as sensitive attributes, and there are 10 categories in total: *Korean, Indian, Mediterranean, Vietnamese, Thai, Chinese, Italian, Japanese, American (New), American (Traditional).*

Baselines. As we are the first to study the quality fairness of personalized text generation, there is no existing work that directly addresses this problem. For comparison, we adapt popular methods in the fairness literature for the purpose, including pre/ in/post-processing methods. Below we briefly introduce them and describe their detailed implementations in Appendix D.

- RAW: Original PETER or NRT model.

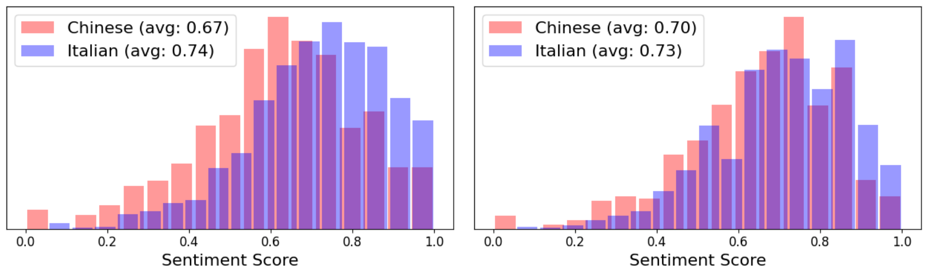


Figure 3.10: The sentiment distributions of generated explanations for Chinese restaurants and Italian restaurants on Yelp. Left is raw PETER generated, and right is generated by WFPTG fine-tuned PETER.

Table 3.5: Results of sentiment fairness on Yelp Restaurants.

Yelp	DP	SDP	BL1	BL4	RG1	RG2	RGL	BERT	Avg-Stm
RAW	0.0739	0.0521	0.1176	**0.0256**	**0.2095**	**0.0452**	**0.1829**	**0.2696**	0.6896
ADV	**0.0322**	0.0410	0.1067	0.0182	0.1971	0.0389	0.1720	0.2337	0.7514
NORM	0.0553	0.0523	0.1256	0.0208	0.1959	0.0392	0.1702	0.2570	0.6778
BT	0.0631	0.0514	**0.1307**	0.0186	0.1799	0.0328	0.1589	0.2629	0.7813
WFPTG	0.0342	**0.0355**	0.1150	0.0195	0.1996	0.0395	0.1751	0.2455	0.7235

Table 3.6: Results of length fairness on Amazon Movies & TV.

Movies	DP	SDP	BL1	BL4	RG1	RG2	RGL	BERT	Avg-Len
RAW	0.3919	0.7529	0.1182	0.0222	0.1673	0.0431	0.1474	**0.1930**	10.98
ADV	0.1835	0.3699	0.1132	0.0194	0.1796	0.0443	0.1535	0.1890	13.22
NORM	0.1809	0.3903	0.1202	**0.0224**	**0.1826**	**0.0507**	**0.1571**	0.1872	13.09
BT	0.1886	0.4334	**0.1225**	0.0206	0.1534	0.0238	0.1294	0.1609	13.47
WFPTG	**0.0587**	**0.1940**	0.1172	0.0221	0.1753	0.0441	0.1512	0.1927	12.29

- ADV (in-processing): Adversarially removing the sensitive information in user or item representations by adding a discriminator [200].
- NORM (pre-processing): Normalizing the training data to remove the bias on group-level.
- BT (pre-processing): Back-translation has been shown to help normalize text and reduce bias [206, 207]. We pre-process the training data by translating the explanations to Chinese and then back to English.
- WFPTG: out proposed method for achieving Wasserstein fairness.

We train each model on the training set, tune the hyper-parameters on the validation set, and report the results on the testing set. Each result point is averaged over 3 runs.

Metrics. Our evaluation consists of two parts: fairness evaluation and utility evaluation. For fairness metrics, we use both the traditional Demographic Parity (DP) which calculates the difference of average measures across groups, and the Strong Dempgraphic Parity (SDP) calcuated by Wasserstein distance between group-wise measure distributions. The utility metrics are the same as used in Section 3.2.5. For Yelp, we consider the fairness of sentiment of generated explanations w.r.t. the restaurant categories. For Amazon Movies & TV, we use the length of explanations to specify the fairness w.r.t. the users' genders.

Results and Analysis

We first visualize the sentiment distributions of explanations generated for Yelp restaurants. We compare the results from WFPTG fine-tuned PETER with the original PETER model in Figure 3.10. As we can see, after fine-tuning with WFPTG, the gap of sentiment distributions for Chinese and Italian restaurants is shrunk. More importantly, the averge sentiments are not changed too much, which still aligns with the original average sentiments. This demonstrates WFPTG's effectiveness in achieving fairness with minimum change on the outputs.

We present the detailed results on fairness and utility in Table 3.5 and Table 3.6 for the sentiment fairness on Yelp data set and length fairness on Amazon Movies & TV data set. As shown in the tables, besides the sentiment fairness evaluated by DP on Yelp, WFPTG achieves superior fairness results on both data sets. On sentiment fairness on Yelp, ADV only outperforms WFPTG on DP with a small edge by pushing the sentiments to be very positive. This shows the generality and effectiveness of WFPTG in achieving fairness in personalized text generation. Moreover, the sentiments or length after WFPTG tuning is close to the original model, meaning that WFPTG can achieve fairness with minimum change on the corresponding measures. Although the utility may drop after fine-tuning by WFPTG, the utility remains largely comparable to the original model. Therefore, WFPTG is promoting fairness efficiently without hurting utility too much.

3.4 Conclusion

In this chapter, we investigate fairness in PS from different perspectives: universal, counterfactual (individual), group-wise fairness. Specifically, we propose a principled new way for learning unbiased node embeddings from graphs biased by sensitive attributes. The idea is to infer a bias-free graph where the influence from sensitive attributes is removed, and then learn the node embeddings from it. This new perspective motivates our design of UGE-W, UGE-R and their combined methods UGE-C. Extensive experiment results demonstrated strong debiasing effect from UGE as well as better unbiasedness-utility trade-offs in downstream applications. For measure-specific fairness, we first study it on individuals from a counterfactual perspective. We propose a general framework called COFFEE for achieving counterfactual quality fairness in explanation generation. Finally, we also consider group-wise fairness in explanation generation with strong demographic parity (SDP). We propose a detailed algorithm WFPTG for achieving SDP by minimizing the Wasserstein distance between measure distributions across groups. Comprehensive results and analysis show the efficacy of our approaches in achieving fairness without hurting the utility of personalized results.

Instead of studying how the generated text may contain representative bias, we firstly study the fairness of personalized text generation in the setting of explanation generation in recommendation. Our work opens the new direction of fairness in personalized text generation, where the target users and items are directly the subjects to the fairness consideration. This problem is general and has more direct and serious implications in real-world applications, since the personalized text will directly affect the user behaviours and decision-making. We expect our work to encourage researchers to investigate novel fairness notions in this problem based on different quality measures, and conduct in-depth analysis on their social impacts. We also plan to generalize the our developed frameworks to other personalized text generation settings, such as conversational systems.

Chapter 4

Conclusion and Future Work

The significance of transparency and fairness in personalization systems (PS) can never be overstated, particularly when the decisions made by these systems can have a direct impact on people's social lives. This dissertation aims to provide a comprehensive understanding of these issues, examining them from multiple perspectives. By incorporating the suggested techniques, a contemporary PS can produce more dependable results when assisting both the consumers and providers in decision-making. Through rigorous analysis and extensive empirical evaluation, we demonstrate the applicability of the proposed methods in diverse contexts and applications.

In Chapter 2, we focus on improving the transparency of PS by providing personalized, intuitive textual explanations that cater to users' preferences. Our goal is to create explanations that are informative, faithful, readable, and comparable. To achieve this objective, we propose several techniques. First, we introduce MTER, which mines the opinionated content in user reviews to generate informative and personalized explanations. To improve the fidelity of explanations, we propose FacT, which combines rule-based decision trees with latent factor models to learn explainable user and item representations. In addition to template-based explanation generation, we explore natural language explanations to increase expressiveness and diversity. To ensure that the sentiment of the explanations aligns with the recommendation results, we develop SAER. Finally, we introduce CompExp, which makes explanations more comparable to assist users in their decision-making process. Extensive experiments demonstrate the effectiveness of our methods, and user studies confirm the practical value of the explanations generated by our approaches.

Chapter 3 examines fairness in PS from various angles, including universal, counterfactual (individual), and group-wise fairness. Our focus is on developing a principled approach to learn unbiased node embeddings from graphs biased by sensitive attributes. The key idea is to infer a bias-free graph by removing the influence of sensitive attributes and learn node embeddings from it. This perspective informs the design of our methods, including UGE-W, UGE-R, and their combined approach UGE-C. Extensive experiments demonstrate that our methods achieve strong debiasing effects and better unbiasedness-utility trade-offs in downstream applications. For measure-specific fairness, we propose the COFFEE framework, which achieves counterfactual quality fairness in explanation generation. We also explore group-wise fairness in explanation generation by incorporating strong demographic parity (SDP). To achieve SDP, we propose a detailed algorithm WFPTG that minimizes the Wasserstein distance between measure distributions across groups. Our results show that our approaches effectively achieve fairness without sacrificing the utility of personalized results.

4.1 Future Work

This dissertation seeks to improve PS by prioritizing transparency and fairness, two critical components for the continued benefit of PS to society. We explore these aspects from different perspectives and propose corresponding solutions. While we present promising methods to address these issues, there are still many challenges that need to be overcome to achieve full transparency and fairness in PS.

4.1.1 Understanding the Dynamics in Providing Explanations in PS

To fully understand the utility of system provided explanations, we need to analyze their effect on the dynamics of user-system interactions. In particular, we can take perspectives stemmed from econometrics studies in *revealed preference theory* [212] to infer user valuation from their repeated interactions with a personalized system. In this case, repeated observations of user actions provide information regarding the choices that were available to those users and which choices were actually taken. We can then recover the utility of explanations by fitting a function of features about the available choices (e.g., price, color, brand of an item) that rationalizes the choices made by the users in the data (expressing their values from different choices). This thus measures the value of explanation, i.e., the *difference* between the utilities of a user's decisions with and without explanation. On top of this, as the explanations would change the dynamics between users and system, the requirement and utility of explanations evolve during the course of interaction. The generated explanations thus should also evolve to provide the most needed information. We can appeal to multi-armed bandit algorithms [213–219] for adaptive explanation generation, where the inferred user valuation of the explained results will be used as feedback for both personalization and explanation.

4.1.2 Analyzing Effects on Users' Decision Making and the Fairness Implications at Large

Understanding how users interact with personalized text and how it influences their decisions is critical to developing fair and unbiased personalized systems. One direction for future work is to investigate the impact of personalized generated text on user behavior and decision-making. Future research could conduct user studies to explore these questions and develop guidelines for designing personalized text that minimizes bias and promotes fairness.

Another area is the development of novel fairness notions for personalized text generation based on different quality measures. We focus on counterfactual and group-wise fairness, but there may be other fairness notions that are more suitable relevant in the context of PS. Developing new fairness notions and evaluating them in real-world applications is a critical step towards building more fair and equitable recommendation systems. From a broader perspective, future research could analyze the social impact of personalized text generation on different communities and demographics. Understanding how different groups are affected by personalized text and how it may exacerbate existing social inequalities is crucial for designing personalized systems that are fair and inclusive. Researchers could conduct large-scale studies to investigate the social impact of personalized text generation and develop guidelines for mitigating any negative effects.

Bibliography

[1] Abhinandan S Das, Mayur Datar, Ashutosh Garg, and Shyam Rajaram. Google news personalization: scalable online collaborative filtering. In *Proceedings of the 16th international conference on World Wide Web*, pages 271–280. ACM, 2007.

[2] Fang Liu, Clement Yu, and Weiyi Meng. Personalized web search for improving retrieval effectiveness. *Knowledge and Data Engineering, IEEE transactions on*, 16(1):28–40, 2004.

[3] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 43–50. ACM, 2005.

[4] Andriy Shepitsen, Jonathan Gemmell, Bamshad Mobasher, and Robin Burke. Personalized recommendation in social tagging systems using hierarchical clustering. In *Proceedings of the 2008 ACM conference on Recommender systems*, pages 259–266. ACM, 2008.

[5] Bin Tan, Xuehua Shen, and ChengXiang Zhai. Mining long-term search history to improve search accuracy. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 718–723. ACM, 2006.

[6] Jaime Teevan, Susan T Dumais, and Eric Horvitz. Personalizing search via automated analysis of interests and activities. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 449–456. ACM, 2005.

[7] Jaime Teevan, Susan T Dumais, and Eric Horvitz. Potential for personalization. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 17(1):4, 2010.

[8] Nan Wang and Hongning Wang. Directional multivariate ranking. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '20, page 85–94, New York, NY, USA, 2020. Association for Computing Machinery.

[9] Yingqiang Ge, Shuchang Liu, Zuohui Fu, Juntao Tan, Zelong Li, Shuyuan Xu, Yunqi Li, Yikun Xian, and Yongfeng Zhang. A survey on trustworthy recommender systems, 2022.

[10] Chris Jay Hoofnagle, Bart van der Sloot, and Frederik Zuiderveen Borgesius. The european union general data protection regulation: what it is and what it means. *Information & Communications Technology Law*, 28(1):65–98, 2019.

[11] Behnoush Abdollahi and Olfa Nasraoui. Using explainability for constrained matrix factorization. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*, pages 79–83. ACM, 2017.

[12] Yongfeng Zhang, M Zhang, Y Zhang, Y Liu, and S Ma. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proc. of SIGIR*, volume 14, 2014.

[13] Mustafa Bilgic and Raymond J Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, volume 5, 2005.

[14] 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. ACM, 2000.

[15] Rashmi Sinha and Kirsten Swearingen. The role of transparency in recommender systems. In *CHI'02 extended abstracts on Human factors in computing systems*, pages 830–831. ACM, 2002.

[16] Flavio P Calmon, Dennis Wei, Bhanukiran Vinzamuri, Karthikeyan Natesan Ramamurthy, and Kush R Varshney. Optimized pre-processing for discrimination prevention. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 3995–4004, 2017.

[17] Tahleen Rahman, Bartlomiej Surma, Michael Backes, and Yang Zhang. Fairwalk: Towards fair graph embedding. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-19*, pages 3289–3295. International Joint Conferences on Artificial Intelligence Organization, 7 2019.

[18] Preethi Lahoti, Krishna P. Gummadi, and Gerhard Weikum. ifair: Learning individually fair data representations for algorithmic decision making, 2018.

[19] Nan Wang, Zhen Qin, Xuanhui Wang, and Hongning Wang. Non-clicks mean irrelevant? propensity ratio scoring as a correction. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, WSDM '21, New York, NY, USA, 2021. Association for Computing Machinery.

[20] Nan Wang, Lu Lin, Jundong Li, and Hongning Wang. Unbiased graph embedding with biased graph observations. In *Proceedings of the ACM Web Conference 2022*, WWW '22, New York, NY, USA, 2022. Association for Computing Machinery.

[21] Meike Zehlike, Francesco Bonchi, Carlos Castillo, Sara Hajian, Mohamed Megahed, and Ricardo Baeza-Yates. Fa*ir: A fair top-k ranking algorithm. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17. Association for Computing Machinery, 2017.

[22] Yunqi Li, Hanxiong Chen, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. User-oriented fairness in recommendation. In *Proceedings of the Web Conference 2021*. ACM, apr 2021.

[23] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Learning fair classifiers. *arXiv: Machine Learning*, 2015.

[24] Moritz Hardt, Eric Price, and Nati Srebro. Equality of opportunity in supervised learning. *Advances in neural information processing systems*, 29:3315–3323, 2016.

[25] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P. Gummadi. Fairness beyond disparate treatment and disparate impact: Learning classification without disparate mistreatment. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17. International World Wide Web Conferences Steering Committee, 2017.

[26] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*, 2018.

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

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

[29] Charu C Aggarwal et al. *Recommender systems*, volume 1. Springer, 2016.

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

[31] Lequn Wang and Thorsten Joachims. User fairness, item fairness, and diversity for rankings in two-sided markets. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21. Association for Computing Machinery, 2021.

[32] Nava Tintarev and Judith Masthoff. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on*, pages 801–810. IEEE, 2007.

[33] Yue Lu, Malu Castellanos, Umeshwar Dayal, and ChengXiang Zhai. Automatic construction of a context-aware sentiment lexicon: An optimization approach. In *Proceedings of the 20th International Conference on World Wide Web*, WWW '11, pages 347–356, New York, NY, USA, 2011. ACM.

[34] Yongfeng Zhang. Incorporating phrase-level sentiment analysis on textual reviews for personalized recommendation. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, WSDM '15, page 435–440, New York, NY, USA, 2015. Association for Computing Machinery.

[35] Amit Sharma and Dan Cosley. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1133–1144. International World Wide Web Conferences Steering Committee, 2013.

[36] William J Clancey. The epistemology of a rule-based expert system7a framework for explanation. *Artificial intelligence*, 20(3):215–251, 1983.

[37] 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.

[38] Aobo Yang, Nan Wang, Hongbo Deng, and Hongning Wang. Explanation as a defense of recommendation. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 1029–1037, 2021.

[39] 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)*. Association for Computational Linguistics, 2021.

[40] Aobo Yang, Nan Wang, Renqin Cai, Hongbo Deng, and Hongning Wang. Comparative explanations of recommendations. In *Proceedings of the ACM Web Conference 2022*, WWW '22. Association for Computing Machinery, 2022.

[41] Nava Tintarev and Judith Masthoff. *Explaining Recommendations: Design and Evaluation*. Springer US, 2015.

[42] H. Schwartz, Maarten Sap, Margaret Kern, Johannes Eichstaedt, Adam Kapelner, Megha Agrawal, Eduardo Blanco, Lukasz Dziurzynski, Gregory Park, David Stillwell, Michal Kosinski, Martin Seligman, and Lyle Ungar. Predicting individual well-being through the language of social media. 01 2016.

[43] H. Samy Alim, John R. Rickford, and Arnetha F. Ball. *Raciolinguistics: How Language Shapes Our Ideas About Race*. Oxford University Press, 2016.

[44] Shikha Bordia and Samuel R. Bowman. Identifying and reducing gender bias in word-level language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*. Association for Computational Linguistics, 2019.

[45] Paul Voigt and Axel von dem Bussche. *The EU General Data Protection Regulation (GDPR): A Practical Guide*. Springer Publishing Company, Incorporated, 2017.

[46] Michelle Seng Ah Lee and Luciano Floridi. Algorithmic fairness in mortgage lending: From absolute conditions to relational trade-offs. *Minds Mach.*, mar 2021.

[47] Ashudeep Singh and Thorsten Joachims. Fairness of exposure in rankings. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '18. Association for Computing Machinery, 2018.

[48] Su Lin Blodgett, Solon Barocas, Hal Daumé III, and Hanna Wallach. Language (technology) is power: A critical survey of "bias" in nlp. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020.

[49] Paul Pu Liang, Chiyu Wu, Louis-Philippe Morency, and Ruslan Salakhutdinov. Towards understanding and mitigating social biases in language models. In *ICML*, 2021.

[50] Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. Societal biases in language generation: Progress and challenges. In *Proceedings of the Conference of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2021.

[51] Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. Counterfactual fairness. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2017.

[52] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 325–333, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR.

[53] Matthew Joseph, Michael Kearns, Jamie H. Morgenstern, Seth Neel, and Aaron Roth. Rawlsian fairness for machine learning. *ArXiv*, abs/1610.09559, 2016.

[54] Chris Russell, Matt J. Kusner, Joshua R. Loftus, and Ricardo Silva. When worlds collide: Integrating different counterfactual assumptions in fairness. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17. Curran Associates Inc., 2017.

[55] Yongkai Wu, Lu Zhang, Xintao Wu, and Hanghang Tong. *PC-Fairness: A Unified Framework for Measuring Causality-Based Fairness*. Curran Associates Inc., 2019.

[56] Karima Makhlouf, Sami Zhioua, and Catuscia Palamidessi. Survey on causal-based machine learning fairness notions, 2020.

[57] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

[58] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. Line: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, page 1067–1077, Republic and Canton of Geneva, CHE, 2015. International World Wide Web Conferences Steering Committee.

[59] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[60] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks, 2017.

[61] Palash Goyal and Emilio Ferrara. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems*, 151:78–94, Jul 2018.

[62] Daixin Wang, Peng Cui, and Wenwu Zhu. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1225–1234, New York, NY, USA, 2016. Association for Computing Machinery.

[63] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1105–1114, New York, NY, USA, 2016. Association for Computing Machinery.

[64] Yongfeng Zhang and Xu Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192:1–101, 2018.

[65] Hanxiong Chen, Xu Chen, Shaoyun Shi, and Yongfeng Zhang. Generate natural language explanations for recommendation, 2021.

[66] Chenhan Yuan and Yi-Chin Huang. Personalized sentence generation using generative adversarial networks with author-specific word usage. *CoRR*, abs/1904.09442, 2019.

[67] 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*, CIKM '18, New York, NY, USA, 2018. Association for Computing Machinery.

[68] Wei-Nan Zhang, Qingfu Zhu, Yifa Wang, Yanyan Zhao, and Ting Liu. Neural personalized response generation as domain adaptation. *World Wide Web*, 22(4), 2019.

[69] Jing Yang Lee, Kong Aik Lee, and Woon Seng Gan. Dlvgen: A dual latent variable approach to personalized dialogue generation, 2021.

[70] Nan Wang, Shaoliang Nie, Qifan Wang, Yi-Chia Wang, Maziar Sanjabi, Jingzhou Liu, Hamed Firooz, and Hongning Wang. Coffee: Counterfactual fairness for personalized text generation in explainable recommendation, 2022.

[71] Mustafa Bilgic and Raymond Mooney. Explaining recommendations: Satisfaction vs. promotion. In *Proceedings of Beyond Personalization 2005: A Workshop on the Next Stage of Recommender Systems Research at the 2005 International Conference on Intelligent User Interfaces*, 2005.

[72] Linas Baltrunas, Bernd Ludwig, and Francesco Ricci. Matrix factorization techniques for context aware recommendation. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, New York, NY, USA, 2011.

[73] Sheng Zhang, Weihong Wang, James Ford, and Fillia Makedon. *Learning from Incomplete Ratings Using Non-negative Matrix Factorization*.

[74] Ruslan Salakhutdinov and Andriy Mnih. Probabilistic matrix factorization. In *20th International Conference on Neural Information Processing Systems*, 2007.

[75] Y. Koren, R. Bell, and C. Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, Aug 2009.

[76] Hao Ma, Haixuan Yang, Michael R. Lyu, and Irwin King. Sorec: Social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM CIKM*, CIKM '08, 2008.

[77] Nan Wang, Hongning Wang, Maryam Karimzadehgan, Branislav Kveton, and Craig Boutilier. Imo^3: Interactive multi-objective off-policy optimization. In Lud De Raedt, editor, *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, pages 3523–3529. International Joint Conferences on Artificial Intelligence Organization, 7 2022.

[78] 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 ACM SIGIR*, pages 83–92, 2014.

[79] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. Social collaborative viewpoint regression with explainable recommendations. In *Proceedings of the Tenth ACM WSDM*, pages 485–494. ACM, 2017.

[80] Tamara G. Kolda and Brett W. Bader. Tensor decompositions and applications. *SIAM Review*, 51(3):455–500, 2009.

[81] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *Proceedings of the Tenth ACM SIGKDD*, KDD '04, 2004.

[82] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Mach. Learn.*, 73(3), December 2008.

[83] Jun Liu, Shuiwang Ji, and Jieping Ye. Multi-task feature learning via efficient l2,1-norm minimization. *CoRR*, abs/1205.2631, 2012.

[84] Chong Wang and David M Blei. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD*, pages 448–456, 2011.

[85] Qiming Diao, Minghui Qiu, Chao Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *ACM SIGKDD*, 2014.

[86] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8), August 2009.

[87] Alexandros Karatzoglou, Xavier Amatriain, Linas Baltrunas, and Nuria Oliver. Multiverse recommendation:n-dimensional tensor factorization for context-aware collaborative filtering. In *ACM Conference on Recommender Systems, Recsys*, 2010.

[88] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. BPR: bayesian personalized ranking from implicit feedback. *CoRR*, abs/1205.2618, 2012.

[89] Wentian Li. Random texts exhibit zipf's-law-like word frequency distribution. *IEEE Transactions on information theory*, 38(6):1842–1845, 1992.

[90] Kamal Amarouche, Houda Benbrahim, and Ismail Kassou. Product opinion mining for competitive intelligence. *Procedia Computer Science*, 73:358–365, 2015.

[91] Soo Min Kim and Eduard Hovy. Identifying opinion holders for question answering in opinion texts. *In Proceedings of AAAI-05 Workshop on Question Answering in Restricted Domains*, 2005.

[92] Ronen Feldman, M Fresko, Jacob Goldenberg, Oded Netzer, and Lyle Ungar. Extracting product comparisons from discussion boards. 8001:469–474, 2007.

[93] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12, 2011.

[94] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. *CoRR*, abs/1602.01585, 2016.

[95] Julian J. McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. Image-based recommendations on styles and substitutes. *CoRR*, abs/1506.04757, 2015.

[96] Ke Zhou, Shuang-Hong Yang, and Hongyuan Zha. Functional matrix factorizations for cold-start recommendation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 315–324, New York, NY, USA, 2011. ACM.

[97] Chris Ding, Tao Li, Wei Peng, and Haesun Park. Orthogonal nonnegative matrix t-factorizations for clustering. In *Proceedings of the 12th ACM SIGKDD*, 2006.

[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*, UAI '09, pages 452–461, Arlington, Virginia, United States, 2009. AUAI Press.

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

[100] Robert Neches, William R. Swartout, and Johanna D. Moore. Enhanced maintenance and explanation of expert systems through explicit models of their development. *Software Engineering, IEEE Transactions on*, (11):1337–1351, 1985.

[101] Michael R Wick and William B Thompson. Reconstructive expert system explanation. *Artificial Intelligence*, 54(1):33–70, 1992.

[102] William van Melle, Edward H Shortliffe, and Bruce G Buchanan. Emycin: A knowledge engineer's tool for constructing rule-based expert systems. *Rule-based expert systems: The MYCIN experiments of the Stanford Heuristic Programming Project*, pages 302–313, 1984.

[103] 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*, WWW '18, pages 1543–1552, Republic and Canton of Geneva, Switzerland, 2018. International World Wide Web Conferences Steering Committee.

[104] Mingxuan Sun, Fuxin Li, Joonseok Lee, Ke Zhou, Guy Lebanon, and Hongyuan Zha. Learning multiple-question decision trees for cold-start recommendation. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining*, WSDM '13, pages 445–454, New York, NY, USA, 2013. ACM.

[105] Shuang-Hong Yang, Bo Long, Alexander J Smola, Hongyuan Zha, and Zhaohui Zheng. Collaborative competitive filtering: learning recommender using context of user choice. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 295–304. ACM, 2011.

[106] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *Journal of Machine Learning Research*, 11(Jan):19–60, 2010.

[107] Mark EJ Newman. Power laws, pareto distributions and zipf's law. *Contemporary physics*, 46(5):323–351, 2005.

[108] 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. ACM, 2018.

[109] James Dougherty, Ron Kohavi, and Mehran Sahami. Supervised and unsupervised discretization of continuous features. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 194–202. Elsevier, 1995.

[110] 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 on World Wide Web*, pages 1583–1592. International World Wide Web Conferences Steering Committee, 2018.

[111] Peng Si Ow and Thomas E Morton. Filtered beam search in scheduling. *The International Journal Of Production Research*, 26(1):35–62, 1988.

[112] Filip Radlinski, Madhu Kurup, and Thorsten Joachims. How does clickthrough data reflect retrieval quality? In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 43–52, New York, NY, USA, 2008. ACM.

[113] 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.

[114] Yiyi Tao, Yiling Jia, Nan Wang, and Hongning Wang. The fact: Taming latent factor models for explainability with factorization trees. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 295–304, New York, NY, USA, 2019. ACM.

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

[116] 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.

[117] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2):1–135, 2007.

[118] Ilya Sutskever, James Martens, and Geoffrey Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1017–1024, 2011.

[119] Chenliang Li, Cong Quan, Li Peng, Yunwei Qi, Yuming Deng, and Libing Wu. A capsule network for recommendation and explaining what you like and dislike. In *Proceedings of the 42nd International*

ACM SIGIR Conference on Research and Development in Information Retrieval, pages 275–284, 2019.

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

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

[122] 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.

[123] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[124] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

[125] Wenyuan Zeng, Wenjie Luo, Sanja Fidler, and Raquel Urtasun. Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*, 2016.

[126] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*, 2015.

[127] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[128] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[129] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[130] Julian McAuley, Jure Leskovec, and Dan Jurafsky. Learning attitudes and attributes from multi-aspect reviews. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, page 1020–1025, USA, 2012. IEEE Computer Society.

[131] 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 &#38; Development in Information Retrieval*, SIGIR '14, 2014.

[132] Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pages 556–562, 2001.

[133] 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.

[134] Kalervo Järvelin and Jaana Kekäläinen. Ir evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, volume 51, pages 243–250. ACM New York, NY, USA, 2017.

[135] 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 on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[136] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *arXiv preprint arXiv:1904.09751*, 2019.

[137] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho, and Jason Weston. Neural text generation with unlikelihood training. *arXiv preprint arXiv:1908.04319*, 2019.

[138] Jason Weston, Emily Dinan, and Alexander H Miller. Retrieve and refine: Improved sequence generation models for dialogue. *arXiv preprint arXiv:1808.04776*, 2018.

[139] Badrul Munir Sarwar, George Karypis, Joseph A Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. *Www*, 1:285–295, 2001.

[140] Marko Balabanović and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, 1997.

[141] Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450, 2018.

[142] Julian McAuley, Rahul Pandey, and Jure Leskovec. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 785–794, 2015.

[143] Tong Chen, Hongzhi Yin, Guanhua Ye, Zi Huang, Yang Wang, and Meng Wang. Try this instead: Personalized and interpretable substitute recommendation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 891–900, 2020.

[144] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence*, pages 452–461. AUAI Press, 2009.

[145] Alexandros Karatzoglou, Linas Baltrunas, and Yue Shi. Learning to rank for recommender systems. In *Proceedings of the 7th ACM Conference on Recommender Systems*, pages 493–494, 2013.

[146] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

[147] Alexander Sasha Vezhnevets, Simon Osindero, Tom Schaul, Nicolas Heess, Max Jaderberg, David Silver, and Koray Kavukcuoglu. Feudal networks for hierarchical reinforcement learning. In *International Conference on Machine Learning*, pages 3540–3549. PMLR, 2017.

[148] Joseph J. Pfeiffer, Sebastian Moreno, Timothy La Fond, Jennifer Neville, and Brian Gallagher. Attributed graph models: Modeling network structure with correlated attributes. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, page 831–842, New York, NY, USA, 2014. Association for Computing Machinery.

[149] Avishek Joey Bose and William L. Hamilton. Compositional fairness constraints for graph embeddings, 2019.

[150] Carol T Kulik, L Robert Jr, et al. Demographics in service encounters: effects of racial and gender congruence on perceived fairness. *Social Justice Research*, 13(4):375–402, 2000.

[151] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 50(1):3–44, 2021.

[152] David Madras, Elliot Creager, Toniann Pitassi, and Richard Zemel. Learning adversarially fair and transferable representations, 2018.

[153] Chirag Agarwal, Himabindu Lakkaraju, and Marinka Zitnik. Towards a unified framework for fair and stable graph representation learning. In *Proceedings of Conference on Uncertainty in Artificial Intelligence, UAI*, 2021.

[154] Enyan Dai and Suhang Wang. Say no to the discrimination: Learning fair graph neural networks with limited sensitive attribute information. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 680–688, 2021.

[155] Maarten Buyl and Tijl De Bie. Debayes: a bayesian method for debiasing network embeddings. In *International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 1220–1229, 2020.

[156] Bo Kang, Jefrey Lijffijt, and Tijl De Bie. Conditional network embeddings, 2018.

[157] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[158] Felix Wu, Amauri Souza, Tianyi Zhang, Christopher Fifty, Tao Yu, and Kilian Weinberger. Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR, 2019.

[159] Andrew Y Ng, Michael I Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[160] Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Nips*, volume 14, pages 585–591, 2001.

[161] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[162] Peng Liu, Lemei Zhang, and Jon Atle Gulla. Real-time social recommendation based on graph embedding and temporal context. *International Journal of Human-Computer Studies*, 121:58–72, 2019.

[163] Min Xie, Hongzhi Yin, Hao Wang, Fanjiang Xu, Weitong Chen, and Sen Wang. Learning graph-based poi embedding for location-based recommendation. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 15–24, 2016.

[164] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[165] Enyan Dai and Suhang Wang. Learning fair graph neural networks with limited and private sensitive attribute information. *arXiv preprint arXiv:2009.01454*, 2020.

[166] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*, 2013.

[167] John Palowitch and Bryan Perozzi. Monet: Debiasing graph embeddings via the metadata-orthogonal training unit, 2020.

[168] Blake Woodworth, Suriya Gunasekar, Mesrob I. Ohannessian, and Nathan Srebro. Learning non-discriminatory predictors. In Satyen Kale and Ohad Shamir, editors, *Proceedings of the 2017 Conference on Learning Theory*, volume 65 of *Proceedings of Machine Learning Research*, pages 1920–1953. PMLR, 07–10 Jul 2017.

[169] Miller McPherson, Lynn Smith-Lovin, and James M. Cook. Birds of a feather: Homophily in social networks. *Review of Sociology*, 27:415–444, 2001.

[170] Timothy La Fond and Jennifer Neville. Randomization tests for distinguishing social influence and homophily effects. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 601–610, New York, NY, USA, 2010. Association for Computing Machinery.

[171] Fan Chung and Linyuan Lu. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences*, 2002.

[172] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *J. Mach. Learn. Res.*, 11:985–1042, March 2010.

[173] T. Kloek and H. K. van Dijk. Bayesian estimates of equation system parameters: An application of integration by monte carlo. *Econometrica*, 46(1):1–19, 1978.

[174] Lubos Takac and Michal Zabovsky. Data analysis in public social networks. In *International scientific conference and international workshop present day trends of innovations*, 2012.

[175] F Maxwell Harper and Joseph A Konstan. The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, 5(4):1–19, 2015.

[176] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[177] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[178] Andriy Mnih and Russ R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pages 1257–1264, 2008.

[179] Matthew Newman, Carla Groom, Lori Handelman, and James Pennebaker. Gender differences in language use: An analysis of 14,000 text samples. *Discourse Processes - DISCOURSE PROCESS*, 05 2008.

[180] Sarah Volz, Marc-André Reinhard, and Patrick Müller. Why don't you believe me? detecting deception in messages written by nonnative and native speakers. *Applied Cognitive Psychology*, 2020.

[181] Sophie Groenwold, Lily Ou, Aesha Parekh, Samhita Honnavalli, Sharon Levy, Diba Mirza, and William Yang Wang. Investigating african-american vernacular english in transformer-based text generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2020.

[182] Annie Louis. Predicting text quality: Metrics for content, organization and reader interest. *Dissertations available from ProQuest*, 2013.

[183] Donald E. Powers. Wordiness: A selective review of its influence, and suggestions for investigating its relevance in tests requiring extended written responses. *Research Memorandum, Educational Testing Service*, 2005.

[184] Liang Guo, Scott A. Crossley, and Danielle S. McNamara. Predicting human judgments of essay quality in both integrated and independent second language writing samples: A comparison study. *Assessing Writing*, 2013.

[185] Johanna Fleckenstein, Jennifer Meyer, Thorben Jansen, Stefan Keller, and Olaf Köller. Is a long essay always a good essay? the effect of text length on writing assessment. *Front Psychol.*, 2020.

[186] Po-Sen Huang, Huan Zhang, Ray Jiang, Robert Stanforth, Johannes Welbl, Jack Rae, Vishal Maini, Dani Yogatama, and Pushmeet Kohli. Reducing sentiment bias in language models via counterfactual evaluation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, November 2020.

[187] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H. Chi, and Alex Beutel. Counterfactual fairness in text classification through robustness. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society*, AIES '19, New York, NY, USA, 2019. Association for Computing Machinery.

[188] Francesco Locatello, Gabriele Abbati, Thomas Rainforth, Stefan Bauer, Bernhard Schölkopf, and Olivier Bachem. On the fairness of disentangled representations. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2019.

[189] Jianxin Ma, Chang Zhou, Peng Cui, Hongxia Yang, and Wenwu Zhu. *Learning Disentangled Representations for Recommendation*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[190] Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Rätsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *ICML*, 2019.

[191] Yu Zheng, Chen Gao, Xiang Li, Xiangnan He, Yong Li, and Depeng Jin. Disentangling user interest and conformity for recommendation with causal embedding. In *Proceedings of the Web Conference 2021*, WWW '21. Association for Computing Machinery, 2021.

[192] Mengnan Du, Fan Yang, Na Zou, and Xia Hu. Fairness in deep learning: A computational perspective. *IEEE Intelligent Systems*, 2021.

[193] Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. A survey on bias and fairness in machine learning. *ACM Comput. Surv.*, jul 2021.

[194] Svetlana Kiritchenko and Saif Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. In *Proceedings of the Seventh Joint Conference on Lexical and Computational Semantics*. Association for Computational Linguistics, 2018.

[195] Jon Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. In *8th Innovations in Theoretical Computer Science Conference (ITCS 2017)*, Leibniz International Proceedings in Informatics. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2017.

[196] Richard Berk, Hoda Heidari, Shahin Jabbari, Michael Kearns, and Aaron Roth. Fairness in criminal justice risk assessments: The state of the art. *Sociological Methods & Research*, 2021.

[197] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, ITCS '12. Association for Computing Machinery, 2012.

[198] Junzhe Zhang and Elias Bareinboim. Equality of opportunity in classification: A causal approach. In *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2018.

[199] Aria Khademi, Sanghack Lee, David Foley, and Vasant Honavar. Fairness in algorithmic decision making: An excursion through the lens of causality. In *The World Wide Web Conference*, WWW '19. Association for Computing Machinery, 2019.

[200] 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*, SIGIR '21, New York, NY, USA, 2021. Association for Computing Machinery.

[201] Shereen Oraby, Lena Reed, Shubhangi Tandon, Sharath T.S., Stephanie Lukin, and Marilyn Walker. Controlling personality-based stylistic variation with neural natural language generators. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*. Association for Computational Linguistics, 2018.

[202] Lei Shu, Alexandros Papangelis, Yi-Chia Wang, Gokhan Tur, Hu Xu, Zhaleh Feizollahi, Bing Liu, and Piero Molino. Controllable text generation with focused variation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, 2020.

[203] Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*, 2020.

[204] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. 2016.

[205] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*. MIT Press, 1999.

[206] Ella Rabinovich, Raj Nath Patel, Shachar Mirkin, Lucia Specia, and Shuly Wintner. Personalized machine translation: Preserving original author traits. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*. Association for Computational Linguistics, 2017.

[207] Jonathan Gabel Christiansen, Mathias Gammelgaard, and Anders Søgaard. The effect of round-trip translation on fairness in sentiment analysis. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2021.

[208] Lily Hu and Yiling Chen. Fair classification and social welfare. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20. Association for Computing Machinery, 2020.

[209] Amanda Coston, Alan Mishler, Edward H. Kennedy, and Alexandra Chouldechova. Counterfactual risk assessments, evaluation, and fairness. In *Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency*, FAT* '20. Association for Computing Machinery, 2020.

[210] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, 2004.

[211] Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020.

[212] Marcel K Richter. Revealed preference theory. *Econometrica*, 34(3):635, 1966.

[213] Joannes Vermorel and Mehryar Mohri. Multi-armed bandit algorithms and empirical evaluation. In *Machine Learning: ECML 2005*, pages 437–448. Springer, 2005.

[214] Jean-Yves Audibert, Rémi Munos, and Csaba Szepesvári. Exploration–exploitation tradeoff using variance estimates in multi-armed bandits. *Theoretical Computer Science*, 410(19):1876–1902, 2009.

[215] Peter Auer, Nicolo Cesa-Bianchi, Yoav Freund, and Robert E Schapire. Gambling in a rigged casino: The adversarial multi-armed bandit problem. In *Foundations of Computer Science, 1995. Proceedings., 36th Annual Symposium on*, pages 322–331. IEEE, 1995.

[216] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670. ACM, 2010.

[217] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. Contextual bandits in a collaborative environment. In *The 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, Pisa, Italy, 2016.

[218] Qingyun Wu, Naveen Iyer, and Hongning Wang. Learning contextual bandits in a non-stationary environment. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, Ann Arbor Michigan, U.S.A., 2018.

[219] Nan Wang, Branislav Kveton, and Maryam Karimzadehgan. Core: Capitalizing on rewards in bandit exploration. In Cassio de Campos and Marloes H. Maathuis, editors, *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, Proceedings of Machine Learning Research. PMLR, 27–30 Jul 2021.

[220] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[221] Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. Multilingual translation with extensible multilingual pretraining and finetuning. 2020.

[222] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2018.

Appendix

A Experimental Settings of UGE

Here we introduce more details about the experiment setup and model configurations for reproducibility.

For GCN-type models (GCN, GAT, SGC), we use two convolutional layers with dimension $d_1 = 64$ and $d_2 = 16$. For node2vec, we set walk length to 1 which turns a general skip-gram loss to objective of the link prediction task. All the deep learning models are trained via Adam optimizer with step size 0.01 for 800 epochs, and we use a normalized weight decay 0.0005 to prevent overfitting. Our proposed UGE methods and the baseline CFC require a regularization weight to balance the task-specific objective and the debiasing effect. For CFC, we report the result with the regularization weight chosen from the set $\{1.0, 5.0, 10.0, 15.0, 25.0, 35.0, 45.0, 55.0, 65.0\}$, which finally is $\lambda = 55.0$. For UGE, we test $\{0.1, 0.3, 0.5, 0.7, 0.9, 1.1, 1.3, 1.5, 1.7, 1.9\}$, and report the performance when $\lambda = 0.5$. The regularization term in Eq (3.1.14) is summed over all node pairs and can be costly to calculate. But empirically, M group pairs sampled uniformly in each round of model update, where M is around 10% of the number of node groups, can already yield promising results. For evaluating the unbiasedness of the node embeddings, we use implementations from scikit-learn [220] for classifier training and evaluating Micro-F1.

B Additional Results of UGE

In Appendix B.1, we include additional experiment results to report the trade-off between unbiasedness and utility on the complete set of embedding models on Pokec-z. In Appendix B.2, we show a complete comparison among our proposed instances of unbiased graph embedding UGE-W, UGE-R and UGE-C. In Appendix B.3, we investigate the influence of the regularization weight on the complete set of embedding models.

B.1 Additional Analysis on Undebiasedness

Table 1 summarizes the debiasing and utility performance of the proposed method and baselines when using four graph neural networks on Pokec-z. Each line of attribute prediction result is followed by the corresponding performance on link prediction. Generally, UGE-W achieves the best link prediction performance and UGE-R has better debiasing effect. Combining UGE-W with UGE-R produces UGE-C with better trade-off.

B.2 Ablation Study

Figure 2 presents the performance of three proposed model (UGE-W, UGE-R and UGE-C) applied to four graph neural networks (GAT, SGC, GCN and node2vec). We can clearly observe that in most cases UGE-R has better debiasing effect compared with UGE-W, while UGE-W can better maintain the utility for downstream link prediction task. UGE-C as the combination of them indeed makes the best of the both designs.

Table 1: The prediction performance of node embeddings learned on Pokec-z using four graph neural networks as embedding models. In each row, we use bold to mark the best debiasedness on attribute prediction or utility on link prediction.

Dataset	Model	Target	No Debiasing	Debiasing Method					Random
				Fairwalk	CFC	UGE-W	UGE-R	UGE-C	
Pokec-z	GAT	Gender (Micro-F1)	0.6232	0.6135	0.5840	0.6150	0.6094	**0.5747**	0.4921
		Link (NDCG@10)	0.3618	0.3280	0.2757	**0.3554**	0.3422	0.3376	0.0570
		Region (Micro-F1)	0.8197	0.8080	0.7217	0.6784	0.7660	**0.6356**	0.4966
		Link (NDCG@10)	0.3618	0.3287	0.2757	0.3451	**0.3547**	0.3098	0.0570
		Age (Micro-F1)	0.0526	0.0522	0.0498	0.0431	0.0545	**0.0429**	0.0007
		Link (NDCG@10)	0.3618	0.3122	0.2757	0.3471	0.3205	**0.3718**	0.0570
	SGC	Gender (Micro-F1)	0.6766	0.6631	**0.6520**	0.6822	0.6531	0.6596	0.4921
		Link (NDCG@10)	0.4975	0.4461	0.4011	**0.4938**	0.4850	0.4765	0.0570
		Region (Micro-F1)	0.7806	0.7820	**0.7150**	0.7402	0.7680	0.7323	0.4966
		Link (NDCG@10)	0.4975	0.4460	0.4011	**0.4832**	0.4799	0.4644	0.0570
		Age (Micro-F1)	0.0621	0.0662	0.0654	0.0606	0.0529	**0.0510**	0.0007
		Link (NDCG@10)	0.4975	0.4461	0.4011	**0.4889**	0.4694	0.4630	0.0570
	GCN	Gender (Micro-F1)	0.5532	0.5589	0.5493	0.5306	0.5301	**0.5162**	0.4921
		Link (NDCG@10)	0.3865	0.2807	0.3836	**0.3851**	0.3727	0.3488	0.0570
		Region (Micro-F1)	0.7445	0.7616	0.7693	0.5800	0.6105	**0.4951**	0.4966
		Link (NDCG@10)	0.3865	0.2807	**0.3836**	0.3801	0.3360	0.3386	0.0570
		Age (Micro-F1)	0.0425	0.0416	0.0391	0.0439	0.0409	**0.0324**	0.0007
		Link (NDCG@10)	0.3865	0.2807	0.3836	**0.3987**	0.3550	0.3391	0.0570
	node2vec	Gender (Micro-F1)	0.5248	0.5347	0.5137	0.5171	**0.4949**	0.4982	0.4921
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.5430	0.5463	0.5206	0.0570
		Region (Micro-F1)	0.8423	0.8462	0.8423	0.8012	0.6490	**0.6372**	0.4966
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.4816	0.5354	0.4506	0.0570
		Age (Micro-F1)	0.0365	0.0404	0.0365	0.0200	0.0122	**0.0068**	0.0007
		Link (NDCG@10)	0.5491	0.5120	**0.5496**	0.5173	0.5439	0.5002	0.0570

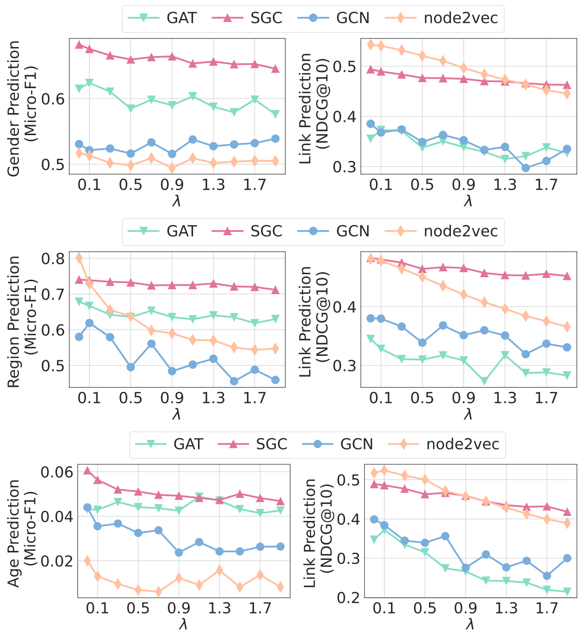


Figure 1: Unbiasedness and utility trade-off using different regularization weights on UGE-C (x-axis). The left columns shows unbiasedness (attribute prediction), and the right columns shows utility (link prediction).

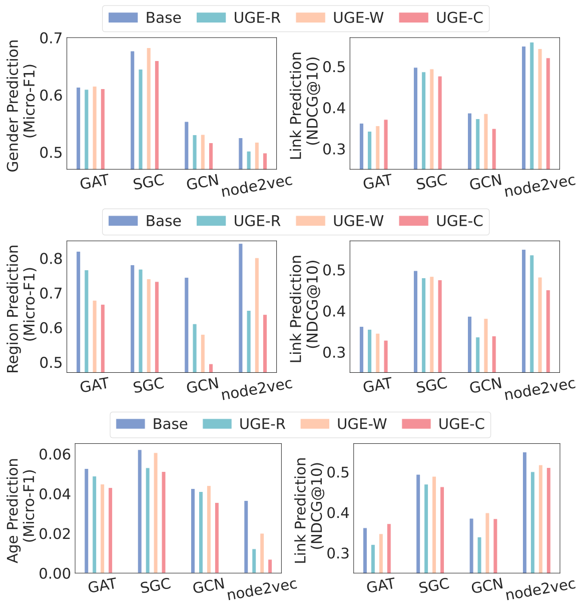


Figure 2: Comparison among our proposed models on different embedding models. The left columns shows the unbiasedness (attribute prediction) and the right columns shows the utility (link prediction).

B.3 Unbiasedness-Utility Tradeoff in UGE

We now include a complete analysis on unbiasedness and utility trade-off across embedding models in Figure 1. It clearly shows a trade-off: as the weight increases, we obtain a stronger debiasing effect with a cost of the utility on link prediction.

C Applying COFFEE to NRT

There are mainly two modules in NRT [37]—a MLP for rating prediction, and an RNN for explanation generation. Both modules share the same user and item embeddings mapped from input user and item IDs for personalization. In particular, the user and item embeddings are used in the initial hidden state of the RNN for explanation generation (see Figure 2 in [37]). To apply COFFEE to NRT, we simply need to use the concatenation of disentangled user preference embedding and the attribute embedding instead of the holistic user embedding, exactly as introduced in Section 3.2.3. Similar to PETER, there are also three loss terms in the original NRT, corresponding to the explanation, context, and rating prediction (see Section 3 in [37]). We denote the total loss of NRT by $\mathcal{L}_{nrt}$, which is used to replace $\mathcal{L}_{gen}$ in eq. (3.2.3) for applying COFFEE.

D Details of Experiment Setup and Additional Results

D.1 Dataset Processing

The first two datasets are the *Video Games* and the *Movies & TV* categories from Amazon reviews. These two datasets only provide the names of users. We use a gender classification tool[1] to predict the gender of users from their names. To guarantee the accuracy of the attribute values, we only reserve the users whose names can be confidently classified as *male* or *female* and remove those classified as *unknown*. In the Yelp dataset, we use a restaurant's price range as the protected attribute. There are originally four price ranges ($, $$, $$$, $$$$) provided. However, we found that there are much less four-dollar restaurants than in the other price ranges, and thus we merge the four-dollar restaurants into the group of three-dollar ones for experiments. Using restaurant price as a sensitive attribute is based on the counterfactual argument: if a restaurant chose to raise/lower the price of its dishes without other changes, the quality of generated explanations shouldn't

[1]https://pypi.org/project/gender-guesser/

change. The rationale is that users may care more about experiences in expensive restaurants since they are paying more, and thus writing more detailed reviews. This would make the explanations generated for expensive restaurants longer and more detailed, which helps recommendations on them be accepted more often, and thus leaves the lower-priced restaurants at a disadvantage.

To construct ground-truth explanations from reviews, we follow the standard processing steps [38, 39, 65] by first extracting item feature words from the all reviews. Then we extract from each review sentences that contain at least one of the feature words as the ground-truth explanation about the user's preferences on the item. We also evaluated RMSE on rating prediction, and the ground-truth ratings range from $1-5$ for all three datasets.

D.2 Details of Baselines

• ADV (in-processing): Adversarially removing the sensitive information in user or item representations by adding a discriminator [200]. We add discriminators on the user's (or item's) embedding in PETER and NRT to remove the information about protected attributes.
• NORM (pre-processing): We normalize the training data to remove the bias on group-level. With two demographic groups, we remove the explanations with the higher (lower) quality in the group with more reviews, until the difference of average quality between the two groups is below 10% of the original difference. With more than two groups, we recursively apply the procedure to the two groups with the maximum difference until the maximum difference is below the threshold. NORM removes less than 2% of training data on Amazon datasets, and less than 18% on Yelp.
• We pre-process the training data by translating the explanations to Chinese and then back to English. We use the multilingual model mBART [221] from EasyNMT[2] to perform the translation.
• ATTR: In order to evaluate the effectiveness of optimizing the fairness constraint in COFFEE, we use the model trained without the constraint ($\lambda = 0$ in eq. (3.2.4)) as a baseline. We call the model ATTR, which means the attribute is disentangled from the user's or item's representations as introduced in Section 3.2.4 but without adding the fairness constraint.
• NATTR (post-processing): We disable the protected attribute token in ATTR for generating explanations during inference. Specifically, for NATTR-NRT, we replace the learned attribute embeddings with random embeddings with values uniformly sampled in $[-1, 1]$ for explanation generation. For NATTR-PETER, we disable the other tokens' attention on the attribute token during inference.

D.3 Evaluation Metrics

We evaluate both individual-level and group-level fairness, with and without counterfactual perspectives. The first metric denoted as Ind-CF evaluates the originally defined CF on individuals [51], which is the same as the regularization term in eq. (3.2.4) averaged over all user-item pairs on the test set $\mathcal{D}$:

$$\begin{aligned}&\text{Ind-CF}\\&=\frac{1}{|\mathcal{D}|}\sum_{u,i\in\mathcal{D}}\left|\mathbb{E}[Q(y_{A\leftarrow a})|u,i]-\mathbb{E}[Q(y_{A\leftarrow a'})|u,i]\right|.\end{aligned}\tag{D.1}$$

We also evaluate the counterfactual effect on group-level [209]. For each attribute value a and the corresponding demographic group, we counterfactually change the attribute value of all users in this group by $a' \neq a$, and calculate the change on the average quality of this group. We call this metric Group-CF:

$$\begin{aligned}&\text{Group-CF}\\&=\frac{1}{|\mathcal{A}|(|\mathcal{A}|-1)}\sum_{a\in\mathcal{A}}\sum_{a'\neq a}\frac{1}{|\mathcal{D}_a|}\\&\left|\sum_{u,i\in\mathcal{D}_a}\mathbb{E}[Q(y_{A\leftarrow a})|u,i]-\sum_{u,i\in\mathcal{D}_a}\mathbb{E}[Q(y_{A\leftarrow a'})|u,i]\right|,\end{aligned}\tag{D.2}$$

[2]https://github.com/UKPLab/EasyNMT

COFFEE	PETER			NRT		
	Games	Movies	Yelp	Games	Movies	Yelp
λ	0.2	0.2	0.2	0.3	0.1	0.1
η	0.6	0.6	0.5	0.5	0.5	0.5

Table 2: Weight λ for the fairness constraint and the promotion weight η in reward calibration of COFFEE when applied to PETER and NRT on the three datasets.

where $\mathcal{D}_a$ denotes the demographic group with attribute value a.

Finally, besides counterfactual metrics, we also evaluate COFFEE's generalization to improve group-wise fairness by the popular notion of demographic disparity (DDP) [23]:

$$\text{DDP} = \frac{2}{|\mathcal{A}|(|\mathcal{A}|-1)} \sum_{a,a' \in \mathcal{A}} \left| \frac{1}{|\mathcal{D}_a|} \sum_{u,i \in \mathcal{D}_a} \mathbb{E}[Q(y)|u,i] - \frac{1}{|\mathcal{D}_{a'}|} \sum_{u,i \in \mathcal{D}_{a'}} \mathbb{E}[Q(y)|u,i] \right|. \quad \text{(D.3)}$$

All fairness metrics are *lower the better*, and the quality expectation on each user-item pair is calculated over $N = 3$ sampled explanations.

D.4 Experiment Settings

Here we elaborate the experiemntal protocols, model specifications, and hyper-parameters. For all models, we set the size of vocabulary to 20,000 by keeping the most frequent words. By default, we use top-5 sampling [222] as the decoding strategy, and the maximum decoding sequence length is 128.

For the raw PETER model [39], we mostly follow the original paper's hyper-parameters. The token embedding dimension is 512 and the dimension of feed-forward network is $2,048$. The number of transformer layers and attention heads are both two. The dropout rate during training is 0.2. For NRT [37], we follow the original paper and set the embedding dimension to 300 for all users, items and words. The dimension of hidden layers is 400. The dropout rate during training is 0.1. COFFEE consists of pre-training the ATTR models with disentangled attribute representations and then fine-tune them with the counterfactual fairness constraints. For ATTR-PETER, we just add an additional attribute token to PETER, with the same embedding dimension for the attribute. For ATTR-NRT, we disentangle the attribute representation by concatenating to the user or item embedding an attribute embedding of dimension 100. The other model specifications for ATTR-PETER and ATTR-NRT are the same as raw PETER and NRT. When applying the discriminator for removing the information about the protected attribute from user or item embeddings, we use a 2-layer MLP with hidden size of 512 as the attribute discriminator. For both ATTR-PETER and ATTR-NRT, the weight λ_D of the adversarial loss is set to 0.5 on Amazon Games and Yelp datasets, and 0 on Amazon Movies. We iterate between one epoch of model training and one epoch of discriminator training until convergence. After the pre-training of ATTR models, we fix the user, item and attribute embeddings, remove the loss on rating prediction but keep the loss on explanation generation, and tune the model with the fairness constraint. The weight λ for the fairness constraint and the promotion weight η in reward calibration are tuned for different models and datasets, and are listed in Table 2.

For model training, we use Adam as optimizer and the initial learning rate for training raw models and ATTR models is 1e-4. The initial learning rate is 1e-5 for COFFEE during fine tuning by the fairness constraint. By default, the batch size is 16. But we use batch size of 8 during the tuning phase of COFFEE when applying to PETER models, mainly due to memory issues. For training raw models and ATTR models, we evaluate the total loss on the validation set after each epoch, and use the epoch checkpoint when the total loss is higher in the next 5 consecutive epochs. When COFFEE fine-tunes the pre-trained ATTR models, we always use the checkpoint after a single epoch, which already yields promising results from COFFEE.

D.5 Unfairness in Explanation Generation

We conduct a case study to understand the fairness issue from a counterfactual perspective. In particular, We study how changing the user's or item's protected attribute counterfactually will affect the length of generated

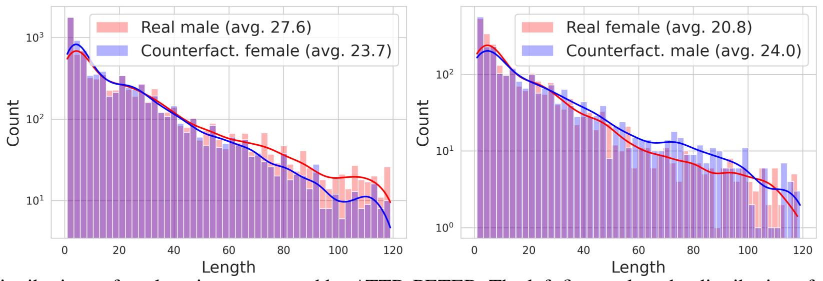


Figure 3: Length distributions of explanations generated by ATTR-PETER. The left figure plots the distributions for real male users vs. their counterpart female users in the counterfactual world. The right figure is similarly plotted for female users vs. their counterfactual male users. The smoothed distribution curves are produced by kernel density estimation.

Table 3: Comparison between COFFEE and baselines based on the PETER model. BL stand for BLEU and RG denotes ROUGE. BLEU, ROUGE and BERTScore are in percentage values and others are in absolute values. The best results are boldfaced, and the second best are underlined. * indicates $p < 0.05$ for significance test over the second best baseline.

PETER	Fairness on Q_{len}			Fairness on Q_{feat}			Utility						
	Ind-CF↓	Grp-CF↓	DDP↓	Ind-CF↓	Grp-CF↓	DDP↓	BL1↑	BL4↑	RG1↑	RG2↑	RGL↑	BERT↑	RMSE↓
Amazon Movies & TV (User's Gender as Protected Attribute)													
RAW	-	-	3.02	-	-	0.57	11.25	**2.20**	18.04	4.53	15.48	19.93	1.11
ADV	-	-	1.43	-	-	0.26	11.08	2.16	17.20	4.32	14.37	18.94	1.14
NORM-L	-	-	1.48	-	-	0.28	10.98	2.11	17.11	4.28	14.78	19.53	1.10
NORM-F	-	-	1.51	-	-	0.27	11.23	2.17	17.46	4.42	15.00	19.67	1.14
BT	-	-	0.94	-	-	0.16	11.25	2.18	12.91	1.66	11.02	15.01	1.13
NATTR	-	-	1.56	-	-	0.27	11.24	2.19	17.35	4.40	14.92	19.34	1.17
ATTR	6.46	0.31	1.87	1.47	0.10	0.37	11.19	2.16	17.43	**4.63**	**15.98**	19.72	1.13
COFFEE-L	3.47	0.07	0.19	1.16	0.01	0.02	10.37	1.52	19.16	4.46	15.88	19.77	**1.08**
COFFEE-F	4.91	**0.06**	0.84	**1.13**	0.01	0.07	**11.28**	2.12	17.77	4.60	15.35	**20.13***	**1.08**
COFFEE-LF	**3.31***	0.16	**0.10***	1.16	0.01	**0.00***	10.05	1.39	**19.34***	4.56	15.86	19.52	**1.08**
Yelp (Restaurant's Price as Protected Attribute)													
RAW	-	-	3.34	-	-	0.51	10.24	1.57	18.71	3.11	14.64	19.82	1.12
ADV	-	-	1.89	-	-	0.28	9.87	1.60	17.25	2.64	13.99	18.34	1.18
NORM-L	-	-	2.09	-	-	0.21	8.57	1.00	**19.38***	**3.27***	**14.76**	19.14	1.18
NORM-F	-	-	0.53	-	-	0.09	9.96	1.54	17.56	2.81	14.10	19.67	1.16
BT	-	-	2.02	-	-	0.27	10.69	1.61	16.42	2.36	13.29	18.89	1.20
NATTR	-	-	1.97	-	-	0.32	10.08	1.55	18.53	3.06	14.57	19.75	1.17
ATTR	9.64	2.65	3.92	1.50	0.44	0.64	10.00	1.53	18.83	3.16	14.75	19.81	1.12
COFFEE-L	2.42	**0.22***	**0.17***	0.61	**0.03**	0.04	11.19	1.64	17.73	2.96	14.49	**20.75***	1.11
COFFEE-F	5.78	0.58	0.71	0.84	0.08	0.13	11.17	**1.77**	17.97	3.00	14.39	20.58	**1.10**
COFFEE-LF	**2.27***	0.31	0.26	**0.54***	**0.03**	**0.03***	**11.61***	1.72	17.41	2.93	14.31	20.21	1.11

explanations, which advocates the needs for CF. We analyze the generated explanations on Amazon Games by ATTR-PETER and the observations are similar on the other two datasets. The analysis is displayed in Figure 3, where we plot the length distributions of generated explanations before and after counterfactually changing users' gender. The left figure corresponds to ground-truth male users and their counterfactual female users. As is shown, when we change male user's gender to female, the length of generated explanations are becoming shorter, with more short explanations and less long explanations. Similarly, in the right figure, the length will become longer if we change ground-truth female users gender to male. These results show that the model generates explanations of different lengths depending on the users' gender, leading to unfair treatment when serving users. Furthermore, this study also proves the effectiveness of disentangling the protected attribute's representations. The separate attribute value input indeed correlates to the quality of generated explanations, laying the foundation for subsequent counterfactual inference for fairness optimization.

D.6 Additional Results

We show the complete results on the Amazon Movies and Yelp datasets based on the PETER model in Table 3. Importantly, COFFEE achieves strong fairness results while maintaining high utility on explanation generation. Although COFFEE may slightly drop the utility compared to the baselines, it can sometimes even outperform the baselines, as shown in the results on Amazon Movies & TV and Yelp datasets. This is because the disentanglement mechanism enables better representation learning [189, 191] and increases the flexibility and accuracy of interactions between the user and item for better explanation generation.

We also present some examples of generated explanations on Amazon Games in Table 4. In particular, we

Table 4: Generated explanations by different models on Amazon Games. We select one male user *Chadwick* and one female user *Noemi*, and present the explanations for them on the same item "*Wii Nunchuk Controller*".

PETER	user: *Chadwick*, item: *Wii Nunchuk Controlle*	user: *Noemi*, item: *Wii Nunchuk Controller*
Ground-truth	this review is for the white wii nunchuk controller. it is a necessary component in most wii games, and attaches to the bottom port of every wii remote. it is well-designed, sturdy, and comfortable to hold. the price is relatively low for such a controller.	arrived as expected.
RAW	i bought it to play wii u and this is a great addition to the wii remote. the gamepad is a nice addition to the wii u gamepad. the wii u is a must have for any wii u console owner.	worked very well.
COFFEE-L	i bought this controller to use the nunchuck. this is a great controller for wii - u owners especially the wii u. the controller is very responsive and the triggers are awesome.	i bought this controller for my wii u. it was exactly what it would be expected. i love the feel of the box and it works great.

select a male user *Chadwick* who wrote long reviews and a female user *Noemi* who wrote short reviews for the same item "*Wii Nunchuk Controller*". The raw PETER model follows the ground-truth reviews and generate detailed long explanation for *Chadwick* but short explanation for *Noemi*. In contrast, after applying COFFEE to improve fairness on length of explanations, the model tend to generate longer and more descriptive explanation for *Noemi*. These examples demonstrate the effect of COFFEE in generating fair and high-quality explanations for users with different protected attribute values.

D.7 Results based on NRT

We present the complete results based on the NRT model in Table 5. Similar to the results on PETER, COFFEE when applied to NRT can significantly outperform the baselines in terms of fairness improvement. Again, when optimizing both Q_{len} and Q_{feat} together, COFFEE achieves the best fairness improvements, verifying the effect of correlations between different quality measures. In general, PETER can generate better explanations than NRT, e.g., on BERTScore, with or without fairness optimizations, showing the capability of transformers over RNNs. However, COFFEE still maintains high generation utility based on NRT when compared to baselines, showing its advantage in generalizing the results to different models.

The observations and conclusions are similar to the PETER based results. These results show COFFEE's ability to generalize the fairness improvement to different types of models, and indicate its flexibility and practicality in real-world uses.

Table 5: Comparison between COFFEE and baselines based on the NRT model. BL stand for BLEU and RG denotes ROUGE. BLEU, ROUGE and BERTScore are in percentage values and others are in absolute values. The best results are boldfaced, and the second best are underlined. * indicates $p < 0.05$ for significance test over the second best baseline.

NRT	Fairness on Q_{len}			Fairness on Q_{feat}			Utility						
	Ind-CF↓	Grp-CF↓	DDP↓	Ind-CF↓	Grp-CF↓	DDP↓	BL1↑	BL4↑	RG1↑	RG2↑	RGL↑	BERT↑	RMSE↓
Amazon Games (User's Gender as Protected Attribute)													
RAW	-	-	7.00	-	-	1.42	8.33	1.30	$\underline{19.21}$	**3.97**	**14.35***	13.79	1.31
ADV	-	-	5.16	-	-	1.25	8.13	1.29	17.67	3.37	13.27	12.85	**1.28**
NORM-L	-	-	7.50	-	-	1.47	8.37	$\underline{1.32}$	$\underline{19.21}$	$\underline{3.92}$	14.27	13.66	1.30
NORM-F	-	-	8.83	-	-	1.79	8.17	1.26	**19.89***	4.27	13.65	13.49	1.30
BT	-	-	8.46	-	-	1.47	8.34	1.17	16.44	2.71	12.23	10.73	$\underline{1.29}$
NATTR	-	-	0.78	-	-	0.12	8.39	**1.39**	13.07	2.71	10.83	11.11	1.30
ATTR	15.54	7.81	8.93	6.70	1.46	1.65	$\underline{8.98}$	1.29	18.94	3.89	$\underline{14.29}$	**13.84**	1.30
COFFEE-L	$\underline{3.74}$	$\underline{0.72}$	$\underline{0.54}$	$\underline{0.90}$	$\underline{0.03}$	**0.03**	**8.99**	1.25	17.65	3.07	12.77	13.82	1.30
COFFEE-F	8.97	3.37	3.47	1.81	0.52	0.51	8.91	1.30	17.94	3.50	13.63	$\underline{13.83}$	1.30
COFFEE-LF	**3.23***	**0.56***	**0.49***	**0.77***	**0.02**	$\underline{0.04}$	8.67	1.26	17.11	3.11	12.46	13.69	1.30
Amazon Movies & TV (User's Gender as Protected Attribute)													
RAW	-	-	2.74	-	-	0.46	10.55	1.86	18.37	3.98	$\underline{15.16}$	19.20	1.06
ADV	-	-	1.56	-	-	0.24	10.48	1.83	$\underline{18.44}$	3.95	15.10	19.09	1.06
NORM-L	-	-	2.74	-	-	0.47	10.27	1.95	17.99	3.81	14.91	19.06	1.06
NORM-F	-	-	2.56	-	-	0.46	10.53	1.91	18.09	3.89	14.96	19.03	1.06
BT	-	-	2.69	-	-	0.43	10.63	1.62	14.84	1.87	11.95	14.36	1.06
NATTR	-	-	1.87	-	-	0.33	9.84	$\underline{2.07}$	15.90	3.37	13.77	16.57	1.07
ATTR	7.75	0.66	2.88	1.74	0.14	0.48	10.53	1.84	**18.68***	**4.10**	**15.34***	19.21	1.07
COFFEE-L	$\underline{4.66}$	$\underline{0.15}$	$\underline{1.38}$	$\underline{1.22}$	0.04	$\underline{0.23}$	**11.06**	$\underline{2.07}$	17.42	3.91	14.90	$\underline{19.29}$	1.07
COFFEE-F	5.75	**0.12**	1.67	1.23	0.04	0.24	$\underline{11.05}$	**2.08**	17.90	$\underline{4.02}$	15.11	19.27	1.07
COFFEE-LF	**4.30***	**0.12**	**1.19***	**1.10***	0.04	**0.19***	10.52	2.02	17.04	3.92	14.77	**19.35***	1.07
Yelp (Restaurant's Price as Protected Attribute)													
RAW	-	-	3.76	-	-	0.56	10.09	1.49	$\underline{19.14}$	**3.17**	$\underline{14.83}$	19.50	**1.01**
ADV	-	-	0.93	-	-	0.21	10.35	1.97	16.78	2.49	13.14	17.88	1.24
NORM-L	-	-	3.21	-	-	0.40	8.76	1.06	18.73	2.87	14.64	18.58	1.07
NORM-F	-	-	1.25	-	-	0.15	9.88	1.42	**19.20**	$\underline{3.14}$	**14.85**	19.48	$\underline{1.06}$
BT	-	-	2.06	-	-	0.30	**10.56***	1.52	16.44	2.31	13.24	18.95	**1.01**
NATTR	-	-	0.54	-	-	0.19	10.38	**2.23***	15.89	2.26	12.58	17.26	1.44
ATTR	10.64	3.19	3.41	1.58	0.42	0.50	10.08	1.85	18.92	3.09	14.72	**19.74**	1.10
COFFEE-L	$\underline{4.17}$	$\underline{0.29}$	**0.19**	$\underline{0.75}$	$\underline{0.12}$	**0.00**	10.29	2.02	17.97	2.87	14.12	19.53	1.10
COFFEE-F	6.88	0.68	0.43	0.93	0.17	0.10	10.25	1.75	18.34	2.92	14.33	19.49	1.10
COFFEE-LF	**3.97***	**0.16***	$\underline{0.21}$	**0.69***	**0.09***	$\underline{0.02}$	$\underline{10.45}$	$\underline{2.07}$	18.65	2.86	14.25	$\underline{19.67}$	1.10