# Synthesizing Natural Language Explanations for Recommendations

---

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Master of Science

by

Aobo Yang

May 2020

# APPROVAL SHEET

This Thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

Author Signature: 

This Thesis has been read and approved by the examining committee:

Advisor: Hongning Wang

Committee Member: Yangfeng Ji

Committee Member: Jundong Li

Committee Member: _____

Committee Member: _____

Committee Member: _____

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

May 2020

# Abstract

Previous research has shown convincing results that explaining the machine-generated recommendations can help customers make more accurate decisions and improve their satisfaction. Many E-commerce applications allow users to leave reviews expressing their experience and sentiments in addition to numerical ratings. These sentimental correlated reviews serve as a perfect resource of explanations for their corresponding ratings. Some works have jointly modeled the recommendation and review generation, but their objective is mostly memorizing the reviews verbatim. In our definition, explanations shall describe features of a given item in an aligned sentiment to defend the recommendation result. Under this insight, we propose a novel neural architecture which not only predicts personalized ratings for recommendation, but also generates supportive explanations describing customized features in a consistent sentiment towards the predicted ratings. On top of a multi-task model, we introduce a rating gate and feature gate to fuse the sentimental representation among the two tasks while specifically emphasizing the sentiment and feature in the generated explanations. Moreover, a content rating supervisor is added in combination with a Gumbel sampler to align the sentiment between end explanations and predicted ratings. To further regularize the generation quality, we adopt adversarial training which can smoothly integrate with the existing Gumbel sampler. Extensive experiments show our model exceeds the baselines in both rating prediction and text generation. Furthermore, the sentimental alignment of generated explanations is significantly improved through our method.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

Personalized recommendation system can improve the efficiency and quality of people's life by providing tailored information according to individuals' preferences. Although a massive amount of research [Herlocker et al., 2004, Koren et al., 2009, Ma et al., 2008] has endeavored to advance the effectiveness of recommendation algorithms, how to explain the machine-made decisions is a fundamental challenge without enough studies. Due to the lack of explainability, many algorithms are black boxes not only to their users but also even to the owners [Zhang and Chen, 2018]. Previous research, however, shows that explanations can help users make more accurate decisions [Bilgic and Mooney, 2005], ease their acceptance of recommendations [Herlocker et al., 2000], and build up confidence towards the system [Sinha and Swearingen, 2002]. Therefore, it is exceptionally necessary to have explainable recommendations which additionally advise users why they should pay attention to the results.

Many E-commerce systems, like Amazon and Yelp, encourage customers to write reviews to share their experiences and feelings to others in addition to just leaving overall sentimental ratings. Users usually read these reviews to figure out the detailed reasons behind given ratings, so reviews is an intrinsic resource for recommendation justification. The correlation between ratings and reviews have largely been utilized to enhance the recommendation quality. Existing explainable recommendation algorithms [Wang et al., 2018, Tao et al., 2019, He et al., 2015] learn to predict the critical elements retrieved from reviews, such as items' features and users' opinionated words.

However, while delivering the end textual explanations, all the methods rely on predefined templates. They require human efforts to prepare one or more templates with blanks to be filled in with predicted aspects or phrases. These semi-customized text snippets lack the expressiveness and diversity of human's natural language. Their strong robotic smell hurts users' trust in the explanations and eventually the entire recommendation system.

As neural network models have achieved extraordinary success in many text generation task recently, such as machine translation [Bahdanau et al., 2014, Luong et al., 2015] and conversational systems [Vinyals and Le, 2015, Li et al., 2016], a few explainable recommendation models [Li et al., 2017b, Lu et al., 2018] also adopt neural network to synthesis natural language explanations. However, they often roughly simplify producing explanations as other text generation tasks, whose main objective is to memorize and reproduce reviews in a verbatim manner. Moreover, they are evaluated mostly by the textual readability in general offline metrics like BLEU [Papineni et al., 2002] and ROUGE [Lin, 2004]. Unfortunately, fairly fluent language does not equal to qualified explanation. Existing works rarely address the unique characteristics required for explanations and specific judgements about how well recommendations are explained. Although some existing works [Ni et al., 2019, Chen et al., 2019] add extra emphasis on the items' features mentioned in explanations, the predicted recommendation ratings or rankings, which are the most essential outputs of the system, are commonly ignored in explanations. The recommendations and explanations are modeled as two loosely linked sub-components with their own separate objectives. We argue this kind of modeling does not capture explanations' actual role which is to elaborate the predicted recommendation results and defend such decisions.

In our vision, explanations shall support items' personalized recommendations in correspondingly aligned sentiments through describing their features. In other words, the sentiments carried within explanation text need to reflect how multiple items are rated or ranked differently by the system and help readers see it. Use Table 1.1 as an example to show the importance of sentiment alignment. Item A's nearly perfect

| Item | Rating | Sentiment Aligned Explanation | Sentiment Unaligned Explanation |
|---|---|---|---|
| *A* | 4.8 | the chicken was so **juicy** and the sauce was **perfect**. | the service is **a little bit slow**. |
| *B* | 2.7 | the burgers were **good**, **but** the fries **weren't that great**. | **great** food, **great** service, **great** atmosphere! |

Table 1.1: Example of recommendations with sentiment aligned and unaligned explanations. The recommendation is modeled as common rating regression problem in scale of one to five. Different sentiments are reflected in the bold words.

rating and item B's below-average one shows the system recommends item A over B. Their sentiment aligned explanations fairly present A is highly recommended (e.g. positive words like "perfect") and B might be less preferred (e.g. transition words "but" and following negation "weren't great"). On the contrary, the unaligned explanations seem to oppose the results since they praise B (e.g. three "great" aspects) while pointing out the flaw of A (e.g. negative word "slow"). The sentiment unaligned explanations will not only fail to assist users but also make them feel confused and further doubt recommendations, which is even worse than no explanation.

Under the above insight, in this work, we propose a novel neural network architecture for sentiment aligned explainable recommendations. Our model is able to give improved recommendations while generating personalized natural language explanations with enhanced feature customization and most importantly, sentiment alignment. Its success is primarily ascribed to two original designs. First, we specifically model the sentiment as a latent vector to bind the tasks of recommendation and explanation together. The sentiment information gets distilled into the vector through back propagation. The idea behind is that since sentiments are shared across both tasks, cooperative learning shall improve sentiments' representations and they will consequently go back to improve both tasks' performance. Since sentiments are directly reflected in end ratings, we model the sentiment vector as a non-linear fusion of user and item representations and feed it as the input to our Multiple Layer

Perceptron (MLP) rater for rating regression. On the other hand, we use a Gated Recurrent Unit (GRU) [Chung et al., 2014] for explanation generation. Intuitively, not every word carries sentiments. For example, in Table 1.1, the stop words like "the" and "of" do not express any attitudes like the bold ones. Therefore, we introduce a sentiment gate and feature gate in the generator to model the sentiment' impact in each position individually. The sentiment gate concept comes from the gates in Long Short Term Memory (LSTM) [Gers et al., 1999] and GRU (e.g. like input gate and forget gate). It calculates a weighted fusion of the generator's hidden state and the sentiment vector to model how much the predicted word should refer to the sentiment. The feature gate is a pointer network [Vinyals et al., 2015, See et al., 2017] of items' features whose attention is calculated with respect to the sentiment vector. The idea has been proven by many previous works [Wang et al., 2018, Wang et al., 2010] that the overall assessment is built upon the feature-level assessments.

The second design is to utilize an independent text sentiment rater to supervise the text generation. This is because the conventional training objective of neural text generation, Maximum Likelihood Estimation (MLE), cannot ensure the aligned sentiment. MLE is a positional loss and treats each word equally so it cannot maintain the sentiment of a whole sentence. Moreover, the generation process in evaluation is different from training. Ground-truth word is fed sequentially in MLE training but previous decoded word is used as the input for the next step [Ranzato et al., 2015]. Most importantly, the generation objective does not involve predicted recommendations at all. The separate recommender and generator may diverge from training in different manners and fail to agree with each other. Therefore, we train a sentiment classifier to predict ratings for the given text. Then use it to guide sentences sampled from the generator to meet the predicted rating. Since the generator output is non-differentiable categorical distribution, we apply Gumbel-Softmax Sampler [Jang et al., 2016] which can estimate the gradient for end-to-end training. To avoid the generation collapses into unreadable and meaningless content merely to please the sentiment supervisor, we build Generative Adversarial Networks (GANs) [Goodfellow et al., 2014] to regularize the text quality. It is smoothly integrated with the

existing Gumbel-Softmax Sampler.

We evaluate our proposed solution on both recommendation performance and explanation quality, which is presented in readability, feature personalization, and sentiment alignment. The experiments are conducted on Yelp dataset with many traditional recommendation baselines and recent neural explainable baselines. Empirical results prove that our model achieves better performance on both tasks. Specifically, our solution can effectively realize sentiment alignment in explanations.

# Chapter 2

# Related Works

Collaborative filtering [Su and Khoshgoftaar, 2009, Schafer et al., 2007] is a widely applied technique to provide recommendations. It leverages historical interactions between users and recommendation items to predict preferences. Latent factor model is its most effective genre. Many such algorithms, like matrix factorization (MF) [Koren et al., 2009], tensor decomposition [Kolda and Bader, 2009], singular value decomposition (SVD) [Koren, 2008], and probabilistic matrix factorization (PMF) [Mnih and Salakhutdinov, 2008], have achieved significant success in recommendation systems. Their essential idea is to compress the interactions into a latent space of lower dimensions. Since neural network shows a promising capacity in latent representation learning due to the nonlinear activation, many deep neural latent models [He et al., 2018, Wang et al., 2015, Zhang et al., 2019] have been proposed for recommendations, such as neural collaborative filtering [He et al., 2017] which combines latent representations of MF and neural network together for preference prediction. Unfortunately, the latent representation and nonlinearity can hardly be interpreted for reasoning. Therefore, the computed recommendation outcomes cannot be questioned.

Lack of reasoning is not a problem only for recommendations. Many general explainable machine learning [Gunning, 2017, Murdoch et al., 2019] research has been conducted to tackle this issue. These solutions [Ribeiro et al., 2016, Lundberg and Lee, 2017] usually focus on system-oriented explanations, explaining the results via tracing connections between the input and output. However, this kind of mathemat-

ically reasonable explanation is more for researchers to deepen their understanding of models rather than for an interactive system like recommendation whose primary goal is to assist end users. Early studies from [Herlocker et al., 2004, Sharma and Cosley, 2013] suggest the necessity of personalized explanation, which focuses on justifying why users should care for the results. Several recent explainable works share the same insight as us. For example, [Wang et al., 2018, Tao et al., 2019] model users' opinions expressed in reviews to generate personalized explanations. However, these methods can only personalize some critical terms, such as features and opinions. Their explanations have to rely on predefined templates that lack the expressiveness and diversity of the nature language. The naive structure and robotic style of such explanations can hurt users' trust in the system.

Neural network models have achieved great success in many text generation tasks, such as machine translation [Bahdanau et al., 2014, Cho et al., 2014], conversational systems [Li et al., 2019, Vinyals and Le, 2015], and text summarization [Rush et al., 2015, See et al., 2017]. Most of the involved techniques, such as recurrent neural network (RNN) language model [Mikolov et al., 2010], attention [Luong et al., 2015], and pointer network [Vinyals et al., 2015], are transparent to the context of generation. Several existing works have exploited the neural network language model within the recommendation system for natural text generation. [Li et al., 2019, Li et al., 2017b] model rating prediction and tip generation together. [Chen et al., 2019] can generate explanations with more accurate features while making recommendations. [Ni et al., 2019] generates recommendation justifications covering different fine-grained aspects. However, these solutions mostly ignore the difference between explanations and other kinds of text that explanations need to elucidate recommendations to assist users in an interactive system. None of them pay attention to the consistent sentiment between recommendation and explanations while leaving the two like independent tasks.

# Chapter 3

# Methods

In this chapter, we will go through the evolvement of our solution. It contains not only the final version of our architecture but also other alternative methods we explored.

The problem of explainable recommendation can be formulated as follows: for a given pair of user $u$ and item $i$, we need to have a model $M$ to output a personalized rating $r$ for recommendation and a sequence of words $w_n$ as explanation. It can be expressed as:

$$r, w_1, w_2, ..., w_n = M(u, i) \tag{3.1}$$

To learn such model, our solution assumes to have a dataset including users, items, ratings, features and explanation sentences. Most recommendation corpora provide raw review documents instead of features and explanations, so preprocessing with other data mining or NLP toolkits [Qi et al., 2020] is necessary. We denote the dataset as $\{U, I, R, F, X\}$ where $U$ is the set of users, $I$ is the set of items, $R$ is the set of ratings, $F$ is the set of features, and $X$ is the set of explanations. We use $r_{u,i}$ and $x_{u,i}$ to denote a pair of rating and explanations in the review of user $u$ and item $i$. We also define a vocabulary set containing all the words $\mathcal{V} = \{w_1, w_2, ..., w_{|\mathcal{V}|}\}$. Since features are items' popular attributes mentioned in the text, it is a subset of vocabulary $F \subset \mathcal{V}$.

## 3.1 Foundation

Because there are two distinct tasks, it is obvious we need a multitask learning model. Inspired by many recommendation algorithms [McAuley and Leskovec, 2013, Almahairi et al., 2015] which have successfully taken advantage of review to improve rating prediction, we believe the necessity to build a shared latent space to capture common sentimental information from both tasks. The initial design thus is given as Figure 3-1: shared user and item embedding bridge the separate rater and text generator. The latent embedding matrices are defined as $P \in \mathbb{R}^{d \times |U|}$ and $Q \in \mathbb{R}^{d \times |I|}$ for user and item respectively, where $d$ is the dimension of the latent vector. For simplicity, they have the same dimensions, but since we always concatenate them in the following work, different dimensions would also work. Given a user $u$ and item $i$, the embedding are written as $p_u$ and $q_i$. The two latent embeddings are concatenated as the input and feed to a multi-layer perceptron network to regress into a numerical rating as the recommendation score. Similarly, the concatenated vector is converted to the initial state of a GRU network to generate a sequence of words as explanations. The losses of the two networks are linearly interpolated as a sum. The whole model is then effectively trained end-to-end with stochastic gradient optimizer of Adam [Kingma and Ba, 2014].

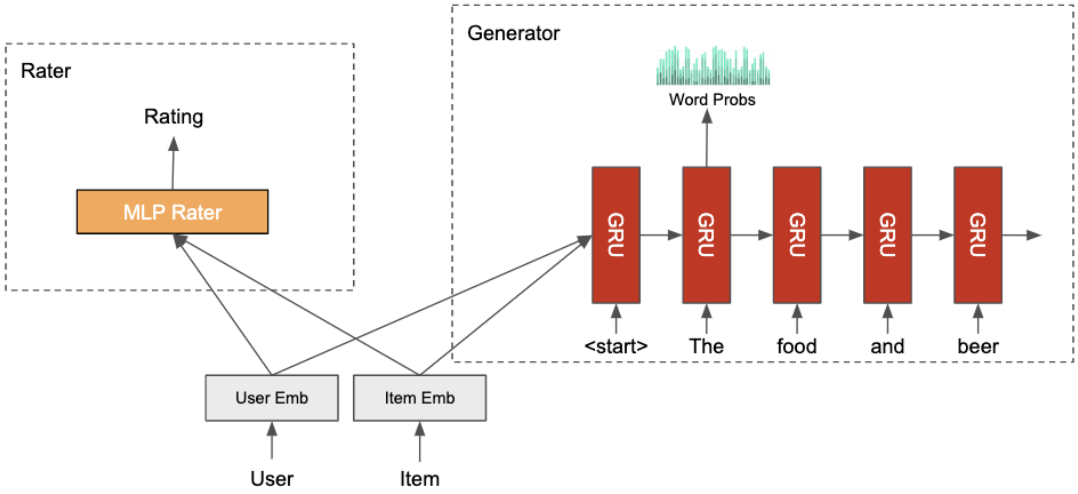


Figure 3-1: Initial multitask learning model with shared user and item embedding

### 3.1.1 Rater

The objective of the rater is to predict a rating based on a pair of user and item, which can be denoted as:

$$\hat{r} = Rater(u, i) \tag{3.2}$$

Inspired by the success of existing neural recommendation models [He et al., 2017], our design uses the popular MLP as the rater which is capable to learn a representation of the complex interaction between a user and item. We are aware that some more complicated add-on or techniques may further boost the performance, but our focus is the whole explainable architecture and the basic MLP is enough for us to demonstrate the improvement.

The rater maps the user $u$ and item $i$ to the corresponding latent embedding at first. Then they are concatenated and passed to the first perceptron layer:

$$h_1^r = relu(W_1^r[p_u, q_i] + b_1^r) \tag{3.3}$$

where $W_1^r \in \mathbb{R}^{d_1^r \times 2d}$ is the weight and $b_1^r \in \mathbb{R}^{d_1^r}$ is the bias of the layer. The superscript $r$ on the right top of symbols means the belonging to the rater and the subscript 1 means the index (e.g. first in the equation) of the layer. The $d_1^r$ is the output hidden vector $h_1^r$'s dimension. We use the symbol $[v_1, v_2]$ to represent the concatenation of vectors of $v_1$ and $v_2$. $relu(\cdot)$ stands for the nonlinear activation function rectified linear unit (ReLU):

$$relu(z) = max(0, z) \tag{3.4}$$

Generally, deeper layers may improve the performance of the prediction. The interactions from the second layer can be abstracted with the layer index $l$:

$$h_l^r = relu(W_l^r h_{l-1}^r + b_l^r) \tag{3.5}$$

where $h_{l-1}^r \in \mathbb{R}^{d_{l-1}^r}$ is the output of the previous hidden layer, $W_l^r \in \mathbb{R}^{d_l^r \times d_{l-1}^r}$ and

$b_l^r \in \mathbb{R}^{d_l^r}$ are the weight and bias of layer $l$.

Assuming the MLP has $L$ layers, the last hidden representation is written as $h_L^r$. To convert it into one rating $\hat{r}$, the rater uses a simple linear layer:

$$\hat{r} = W_r^r h_L^r + b_r^r \tag{3.6}$$

where $W_r^r \in \mathbb{R}^{1 \times d_L^r}$ and $b_r^r \in \mathbb{R}^1$.

The objective is to predict the ground truth rating $r$, so we apply Minimal Squared Error (MSE) as the loss:

$$L^r = \frac{1}{|R|} \sum_{r_{u,i} \in R} (\hat{r}_{u,i} - r_{u,i})^2 \tag{3.7}$$

where $R$ is the set of ratings and $r_{u,i}$ is the rating of user $u$ gives to item $i$. Recommendation, however, is a ranking problem in the end. Minimizing element-wise prediction error does not necessarily help differentiate the relative relevance quality of a set of items. Therefore, we introduce the Bayesian Personalized Ranking (BPR) [Rendle et al., 2012] to enhance the ranking performance. Specifically, we prepare a set of personalized item pairs:

$$B = \{(u, i, j) | r_{u,i} > r_{u,j}\} \tag{3.8}$$

which $i$ and $j$ are two items rated by user $u$. $r_{u,i} > r_{u,j}$ requires item $i$ being rated higher than $j$. The original BPR is evaluated on data of implicit feedback, such as clicked vs unclicked, so it only requires one recorded item in a pair and relies on negative sampling for the counterpart. Our scenario assumes having explicit user feedback to differentiate sentiments. Thus, we revise BPR on pairs of both rated items. With set $B$, BPR ranking loss can be realized by:

$$L^b = -\frac{1}{|B|} \sum_{(u,i,j) \in B} \log \sigma(\hat{r}_{u,i} - \hat{r}_{u,j}) \tag{3.9}$$

The $\sigma(\cdot)$ is the sigmoid function:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.10}$$

The physical meaning of $\sigma(\hat{r}_{u,i} - \hat{r}_{u,j})$ is the estimated probability of item $i$ is more preferred than $j$ for user $u$. Hence, the BPR loss can also be viewed as the negative log likelihood (NLL) loss.

### 3.1.2 Explanation Generator

The responsibility of the explanation generator is to translate the given user and item embedding into a sentence, a sequence of words:

$$\hat{x} = \hat{w}_1, \hat{w}_2, ..., \hat{w}_n = Generator(u, i) \tag{3.11}$$

To solve this problem, we need to model the conditional probability of the target text. Currently, the most popular method is Recurrent Neural Network (RNN). The text likelihood is viewed as the product of a sequence of word probabilities conditioned on the given context, which includes all the previous words, and user and item in our case, so the problem can be formulated as:

$$P(x|u, i) = \prod_{w_t \in x} P(w_t|u, i, w_1, ..., w_{t-1}) \tag{3.12}$$

Since each word is a discrete category of the vocabulary, the goal of each timestamp $t$ becomes a classification problem which produces a probability distribution of the vocabulary:

$$P(w_t|u, i, w_1, ..., w_{t-1}) = y_t(w_t) \tag{3.13}$$

where $y_t$ is the word distribution at position $t$.

As shown on the right side of the Figure 3-1, we apply GRU to model the word distribution $y_t$. The initial context only includes the user and item, so we concatenate the mapped latent embedding of the two and linearly convert it to the initial hidden

state:

$$h_0^x = W_h^x[p_u, q_i] + b_h^x \tag{3.14}$$

where $W_h^x \in \mathbb{R}^{d_h^x \times 2d}$ and $b_h^x \in \mathbb{R}^{d_h^x}$. The superscript $x$ means the variables relate to explanation generation. The $d_h^x$ is the dimension of GRU's hidden state.

For each timestamp, the hidden state shall be updated by a input word, so we define another latent matrix for the word embedding $V \in \mathbb{R}^{d_v^x \times |\mathcal{V}|}$, where $d_v^x$ is GRU's input dimension. $v_t$ is used for the embedding vector of word $w_t$ at timestamp $t$. We initialize $V$ with the pretrained GloVe [Pennington et al., 2014]. The job of GRU is to generate hidden state $h_t^x$ at timestamp $t$ with previous state $h_{t-1}^x$ and input $v_t$:

$$h_t^x = GRU(h_{t-1}^x, v_t) \tag{3.15}$$

Its detailed calculations can be briefly expanded as:

$$\begin{aligned}
r_t^x &= \sigma(W_{vr}^x v_t + b_{vr}^x + W_{hr}^x h_{t-1}^x + b_{hr}^x) \\
z_t^x &= \sigma(W_{vz}^x v_t + b_{vz}^x + W_{hz}^x h_{t-1}^x + b_{hz}^x) \\
n_t^x &= tanh(W_{vn}^x v_t + b_{vn}^x + r_t^x * (W_{hn}^x h_{t-1}^x + b_{hn}^x)) \\
h_t^x &= (1 - z_t^x) * n_t^x + z_t^x * h_t^x
\end{aligned} \tag{3.16}$$

where $*$ is the Hadamard product, $r_t^x$ is the reset gate, and $z_t^x$ is the update gate.

To get the final word probability distribution $y_t$, we pass the hidden state $h_t^x$ through the output layer:

$$y_t = softmax(W_y^x h_t^x + b_y^x) \tag{3.17}$$

where $W_y^x \in \mathbb{R}^{|\mathcal{V}| \times d_h^x}$ and $b_y^x \in \mathbb{R}^{|\mathcal{V}|}$ linearly transform the hidden state into a vector of size of the vocabulary. Each dimension of the vector is a word's logit and the $softmax$ function converts the logit into the probability:

$$softmax(z)_k = \frac{e^{z_k}}{\sum_{j=1}^{|z|} e^{z_j}} \tag{3.18}$$

where $z = [z_1, z_2, ..., z_{|z|}]$ and the subscript $k$ is the dimension index.

Since the objective is to maximize the likelihood of the ground truth explanations, we adopt NLL loss:

$$L^x = -\sum_{x \in X} \sum_{w_t \in x} \log y_{x,t}(w_t) \tag{3.19}$$

In the testing stage, there is no more ground truth explanation $x$ serving as the input, so a decoding/searching strategy is needed to select a word $\hat{w}_t$ at the current timestamp for the next timestamp. Greedy decoding and sampling are the two most common methods. The greedy decoding chooses the word with the highest probability in $y_t$:

$$\hat{w}_t = \arg\max_{w_j \in W} y_t(w_j) \tag{3.20}$$

The sampling strategy randomly picks a word in respect to the multinomial distribution $y_t$. We use sampling for our model due to two reasons: first, greedy decoding tends to produce less diversified safe contents, which is against our idea of personalized explanation; second, a review may contain more than one explanation sentence and the randomness in sampling allows us to generate multiple sentences covering different features through iteration.

### 3.1.3 Multitask Learning

With the three losses from the two tasks, we can conclude one final loss function to train the whole model end-to-end:

$$L = \lambda_r L^r + \lambda_b L^b + \lambda_x L^x + \lambda_n ||\Theta||^2 \tag{3.21}$$

where $L^r$, $L^b$, and $L^x$ are the rating MSE loss, ranking BPR loss, and generation NLL loss respectively. Additionally, we also append the L2 regularization for the model parameters $\Theta$ which includes user embedding $P$, item embedding $Q$, word embedding $V$, and all other variables (all weights $W$ and bias $b$). The $\lambda$ are the corresponding weights for these terms.

## 3.2 Sentiment Modeling

Intuitively, the same sentiment is shared in a user's actions of rating an item and writing its review. While the rating is basically a numerical representation of the sentiment, the review is its comprehensive textual expression. In the previous section, we propose a model that utilize a shared latent user and item space to learn from the back propagation of both tasks. Ideally, the embedding should thus be more representative and expressive for sentiments. Unfortunately, our preliminary test shows no obvious improvement in either task. The separate sub-models with their own embedding achieve the same performance as our combined multitask model. Therefore, we believe the previous basic multitask model is not enough to capture the shared sentiment information. Specifically, the generator is the bottleneck. Because text generation depends on many factors other than sentiments, such as vocabulary preference and syntactic structure, pushing these factors to the shared latent embedding may dilute the concentration on sentiments. For example, in Table 1.1, the same embedding needs also to memorize how to correctly produce stop words, like "the" and "was", which distract the learning of sentimental words, like "great". This theory can also explain why previous works [Wang et al., 2018] using extracted feature-opinion phrases are able to claim improved performance.

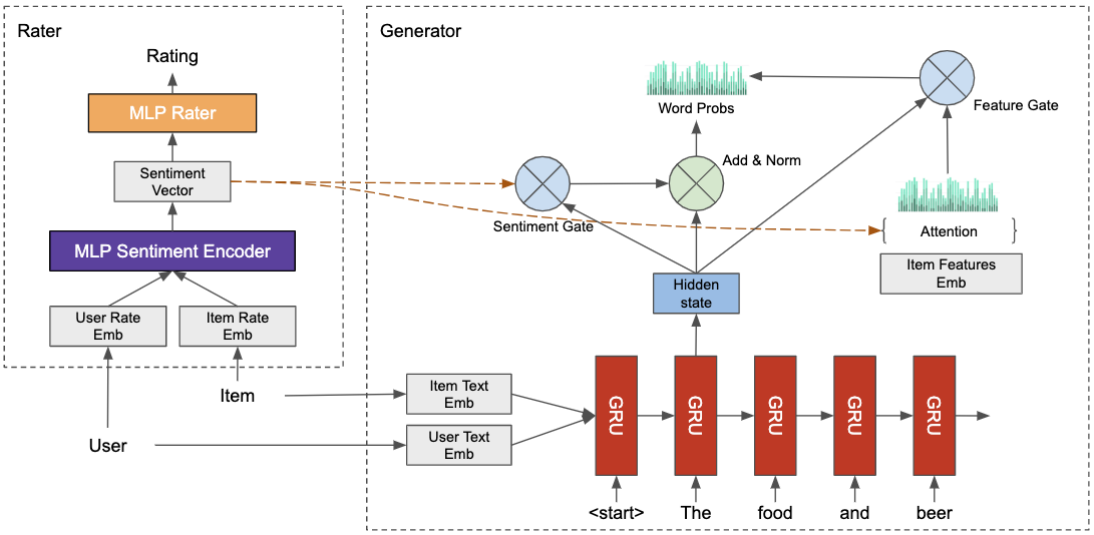


Figure 3-2: Multitask learning model with enhancement on sentiment modeling

According to the above insight, we propose an enhanced multitask model that specifically focuses on sentiment modeling. The architecture is shown in Figure 3-2. It contains four refinements on top of the previous design. First, instead of a shared set of embedding, the Rater and Generator have their own sets of user and item embedding respectively. As we already mentioned, the text generation contains many factors unrelated to sentiments, so with shared embedding, the Rater actually needs to extricate sentiments from the noisy embedding. Separate embedding makes the responsibilities more straightforward. Second, an MLP sentiment encoder is added in the Rater to learn a latent sentiment representation. We define the output latent vector as sentiment vector and use the previous MLP rater to translate it into the numeric rating. Since both the sentiment encoder and rater are MLP, this refinement can also be viewed as cutting the original MLP into two parts and taking the latent vector in the boundary to represent sentiments. Since we know the sentiment impacts the explanations while the disentangled embedding breaks the linkage between the Rater and Generator, the sentiment vector serves as the new shared latent space for both tasks. Third, we introduce a soft gate named sentiment gate in the generator to fuse the sentiment vector into the generation process. The theory behind is that not all words reflect sentiments, so our design allows position-wise decision. Moreover, sentiments work very subtly in neural language: intuitively, it may impact transition words, negation words, adjectives and more. Therefore, no one can craft any hard rules to categorize sentiment words to give direct supervision. Thus, we apply a soft gate learned from GRU and LSTM (e.g. reset gate, input gate) to learn the relevance of each position. At last, we add the feature gate to enhance the mutual learning of sentiments and feature personalization. The idea bases on the correlation between overall ratings and feature/aspect ratings [Wang et al., 2018, Wang et al., 2010]. The feature gate is a pointer network (or copy mechanism) [See et al., 2017, Zeng et al., 2016] using sentiment vector to weight feature words of the specified item. It is named "gate" because the pointer network also includes a soft gate to predict the copy probability.

In summary, our model uses sentiment vectors as the media to distill common

knowledge from related sub-tasks, which includes rating prediction, sentimental words generation, and customized feature prediction, to learn improved representations.

### 3.2.1 Disentangled Embedding

Inspired by many disentanglement works in NLP area [Bao et al., 2019, Wang et al., 2019], we feel the necessity of having disentangled sentiment and textual embedding for different objectives. We now split the original set of user and item embedding $P$ and $Q$ into two separate sets for the rater and generator respectively. Following the naming convention, we use superscript $r$ and $x$ to differentiate them. $P^r \in \mathbb{R}^{d^r \times |U|}$ and $Q^r \in \mathbb{R}^{d^r \times |I|}$ are the user and item rating latent matrices. $P^x \in \mathbb{R}^{d^x \times |U|}$ and $Q^x \in \mathbb{R}^{d^x \times |V|}$ are the user and item text latent matrices. This also applies to embedding vector. For example, $p_u^r$ stands for the rating user embedding of user $u$.

### 3.2.2 Sentiment Encoder

The sentiment encoder is a MLP network which outputs the sentiment vector by simulating the nonlinear interaction between user and item. It is responsible to learn a decent representation about sentiment. It takes the concatenated user and item rating as input to calculate the first latent vector:

$$h_1^s = relu(W_1^s[p_u^r, q_i^r] + b_1^s) \tag{3.22}$$

where the superscript $s$ marks the variables of the sentiment encoder. To increase the model capacity, the encoder has more than one layer. Similar to Equation 3.5, the latent output after layer one can be abstracted as:

$$h_l^s = relu(W_l^s h_{l-1}^s + b_l^s) \tag{3.23}$$

where $h_l^s \in \mathbb{R}^{d_l^s}$. Assume the encoder has $L$ layers, the last latent vector is written as $h_L^s$. The $h_L^s$ is the sentiment vector shown in Figure 3-2. The sentiment vector

becomes a critical input to the downstream models, so we simplify it as:

$$s = h_L^s \tag{3.24}$$

The changes to the MLP rater are straightforward. Instead of directly consuming user and item embedding, it passes the learned sentiment vector through its first layer. Therefore, we replace the Equation 3.4 with:

$$h_1^r = relu(W_1^r s + b_1^r) \tag{3.25}$$

As long as we fix its first layer, everything else shall stay the same.

### 3.2.3 Sentiment Gate

Sentiment gate is a mechanism we bring in to imitate how sentiments play in natural writing: people tend to stop at certain positions to choose the most appropriate words to express their feelings while unconsciously following their wording habits to continue the expression in the other positions. This idea supports our intuition that not all words reflect sentiments. Our design forces the model to learn the sentimental positions. The sentiment gate calculates a percentage $g_t^x$ with respect to GRU's hidden state $h_t^x$. The sentiment vector is weighted by the percentage and then merged with the hidden state to predict word distributions. Its physical meaning is that given the current textual context, how much the next word should refer to the sentiment.

One existing popular method of sentiments modeling in text generation is to dump sentimental representations in the initial state of the generator [Li et al., 2019, Li et al., 2017b]. However, this way makes the prediction of each word depending on sentiments. Also similar to our previous design, the sentiment vector may receive too many noisy back propagation from unrelated words. On the contrary, our sentiment gate is able to mitigate the noise in both directions.

To elaborate how sentiment gate works, we need to re-address how the hidden state $h_t^x$ is calculated from GRU with the new disentangled embedding. The basic

flow is actually the same except the initial state is calculated with the text specific embedding:

$$h_0^x = W_h^x[p_u^x, q_i^x] + b_h^x \tag{3.26}$$

Then as before:

$$h_t^x = GRU(h_{t-1}^x, v_t) \tag{3.27}$$

Since $h_t^x$ is supposed to capture previous textual context, we use it to determine how much the next word shall refer sentiment. The steps is quite similar to the detailed steps of GRU shown in Equation 3.16. We first calculate the sentiment gate $g_t^x$. Then use $g_t^x$ to weight sentiment vector $s$ and merge it back with the hidden state.

$$\begin{aligned} g_t^x &= \sigma(W_g^x h_t^x + b_g^x) \\ m_t^x &= tanh(h_t^x + g_t^x * (W_m^x s + b_m^x)) \end{aligned} \tag{3.28}$$

where $W_g^x \in \mathbb{R}^{1 \times d_h^x}$ and $b_g^x \in \mathbb{R}$ produce a scalar $g_t^x$ not vector. $m_t^x$, the new hidden state fused with sentiment, replaces the original $h_t^x$ in the output layer to predict the word distribution:

$$y_t^m = softmax(W_y^x m_t^x + b_y^x) \tag{3.29}$$

The $y_t^m$ is given an additional superscript $m$ because it is no longer the final word distribution due to the feature gate. We will introduce feature gate in the next section.

### 3.2.4 Feature Gate

The objective of feature gate is to improve the delivery of accurate feature words in explanations meanwhile encode feature preference in the sentiment vector to sharp the rating prediction. The feature gate is essentially a pointer network which copies the item's features into the end word distribution. We divide its function into two steps. First, it calculates the sentiment vector $s$'s attention towards the features, whose physical meaning is the customized feature preference. Second, it uses textual hidden state $h_t^x$ to calculate a ratio $c_t^x$ to linearly interpolate the word distribution $y_t^m$

and feature attention to have the final word distribution $y_t$, whose physical meaning is that given the current textual state, how likely to mention an item's feature next.

Since the targets are item specific feature words, for each item $i$, we build a feature set $F_i = \{f_k | f_k \in \{x_{u,i} | u \in U\}\}$ which means it contains every feature word that appears in any item $i$'s training data. Features are also words, so we simply reuse the word embedding $V$ to represent features. We apply the method "general" in [Luong et al., 2015] to calculate the attention:

$$\begin{aligned} z_{t,k} &= [h_t^x, s]^T W_z^x v_{f_k} \\ a_t &= softmax(z_t) \end{aligned} \tag{3.30}$$

where $z_{t,k}$ is computed for every $f_k$ in $F_i$ and is the dimension $k$'s value of vector $z_t$. $a_t$ is attention distribution and we use $a_t(f_k)$ to indicate the ratio of $f_k$. The attention's input contains the hidden state $h_t^x$ in addition to the sentiment vector $s$ because the textual context may help exclude some semantic incorrect or duplicate features. For better performance, an extra linear transformation can be applied to $h_t^x$ to compress it into lower dimension, which help avoid overfitting attentions to the text and ignoring the sentiment.

To decide if the model shall copy features based on the attention, we obtain the copy probability by:

$$c_t^x = \sigma(W_c^x h_t^x + b_c^x) \tag{3.31}$$

where $W_c^x \in \mathbb{R}^{1 \times d_h^x}$ and $b_c^x \in \mathbb{R}$. The $c_t^x$ allows us to mix the generation and feature attention to get our final word distribution:

$$y_t = (1 - c_t^x) y_t^m + c_t^x a_t \tag{3.32}$$

where $a_t(w_k)$ is 0 if $w_t \notin F_i$.

## 3.3 Sentiment Supervision

So far, we have modeled how the generator should use sentiments in explanation generation, but it is still not enough to guarantee the sentiment alignment. There are several reasons. First, word-based NLL training cannot maintain the whole sentence's sentiment. Second, the generation process in testing works differently with the training that there are no ground truth words, so it is safe to claim the actual workflow has never been trained. Third, the end rater and generator may parse the same sentiment vector differently. In other words, since the predicted rating can diverge from the factual rating, even the generator can perfectly produce the ground truth sentence, it is no longer a supportive explanation to the predicted recommendation. In summary, the objectives of rater and generator are disconnected. They never realize the existence of each other. Therefore, we introduce a sentiment alignment loss to bind them. It directly trains the testing-like sampled sentence as a whole to agree with the predicted rating.

The final version of our architecture is shown in Figure 3-3. On top of the multitask model, we create a new supervisor module. The content rater is the critical component that brings the ultimate output of the rater and generator together. Its role is like a judge which evaluates the sentimental rating of the generated explanation. We pretrain an independent regression model $CR$ which predicts the ground truth rating $r$ according to the ground truth explanation $x$. Then we freeze it and use it to predict the rating of the generated explanation $\hat{r}^x$. In order to provide guidance to the generator, we apply MSE loss to push the content rating $\hat{r}^x$ to meet the predicted rating $\hat{r}$. As we have already emphasized, explanations are the defense of the recommendation system's decisions. So note the target label is the predicted rating $\hat{r}$ instead of the ground truth rating $r$. However, it is not easy to pass updates back to the generator since the words sampling process cannot be differentiated. We tried two existing solutions. One is to cast the problem to a reinforcement learning scenario and apply policy gradient [Williams, 1992, Glynn, 1990]. We explore two particular implementations: the Minimum Risk Training used in [Shen et al., 2015, Ayana et al.,

2016]; the Monte Carlo (MC) search used in [Yu et al., 2017]. Unfortunately, we find them significantly time-consuming [Li et al., 2017a] and suffering high variance. The other solution, which is also the one in our final model, is to employ Gumbel Softmax [Jang et al., 2016] to estimate the gradient of sampling from categorical distribution. Its training is both efficient and stable.

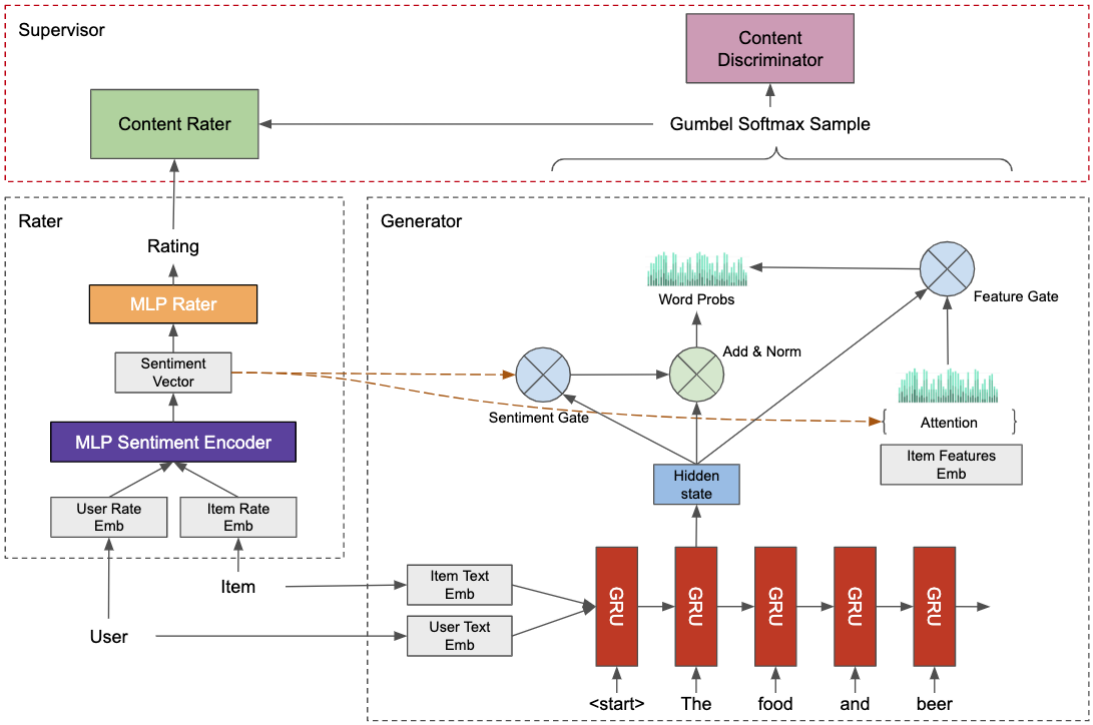


Figure 3-3: Final model architecture with sentiment supervisor

Besides the content rater, the supervisor module also applies adversarial learning to regularize the truthfulness of generated explanations. It effectively helps avoid the generation being trapped in producing meaningless content. The reason behind the problem is that the generator may be rewarded for producing unreadable sequences of words which however please the content rater. As a regression model, the content rater rates every input including out-of-distribution ones. For example, it may give a very positive rating to an unnatural sentence "good good good good ...". Such rating is meaningless so we need to punish the generator for doing so. GAN is potent for realistic generation. We add a content discriminator $CD$ to classify the authentic and generated explanations so that we can simultaneously train the generator to produce

convincing content to fool it.

### 3.3.1 Content Rater

Our architecture gives no assumption to the algorithms of the content rater. As long as it can convert text into a numerical score, it should work. However, it needs to have a decent performance to make the supervision reliable. We expect it to perform better in rating metrics like RMSE on ground truth explanations than the user and item rater. The intuition is that reading the actual explanations should understand the sentiment better than guessing with the user and item's rating history. In our implementation, the content rater consists of a bidirectional RNN text encoder with inner attention and a MLP on top for rating regression. The process is formulated as:

$$\hat{r}^x = CR(\hat{x}) \tag{3.33}$$

where $\hat{x}$ is the decoded explanation. The objective is to align the content rating $\hat{r}^x$ with the predicted rating $\hat{r}$, so we employ MSE loss across all possible $\hat{x}$:

$$\begin{aligned} L^{cr} &= \sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[(\hat{r}^x - \hat{r})^2] \\ &= \sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[(CR(\hat{x}) - \hat{r})^2] \end{aligned} \tag{3.34}$$

where $\mathbb{E}[\cdot]$ stands for expectation and $P(\hat{x}|u,i)$ is the probability of sampling $\hat{x}$ with given $u$ and $i$. However, due to the categorical sampling process, the loss' gradient cannot pass to the generator through $\hat{x}$. We will discuss two solutions in the next sections.

### 3.3.2 Policy Gradient

The policy gradient views the sampling probability $P(\hat{x}|u,i)$ of our multitask model as the policy and the rating alignment loss in supervisor as the action value. The objective is to tune the parameters in the multitask model to improve the policy

to maximize the reward value or minimize the risk value. In our case, since we try to minimize the content rating loss $L^{cr}$, so we define the risk value as:

$$\Delta(\hat{x}, \hat{r}) = (CR(\hat{x}) - \hat{r})^2 \tag{3.35}$$

We use $\Theta$ to represent all the other variables in our multitask model. The user and item rater is frozen in the supervision training, which means we do not update the predicted rating $\hat{r}$, so $\Delta(\hat{x}, \hat{r})$ does not impact $\Theta$. Then with the logarithmic trick, we can compute $\Theta$'s gradient as:

$$\begin{aligned}
\nabla_\Theta L^{cr} &= \nabla_\Theta \sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[\Delta(\hat{x}, \hat{r})] \\
&= \sum_{u,i} \sum_{\hat{x}} \nabla_\Theta P(\hat{x}|u,i) \cdot \Delta(\hat{x}, \hat{r}) \\
&= \sum_{u,i} \sum_{\hat{x}} P(\hat{x}|u,i) \nabla_\Theta log P(\hat{x}|u,i) \cdot \Delta(\hat{x}, \hat{r}) \\
&= \sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[\nabla_\Theta log P(\hat{x}|u,i) \cdot \Delta(\hat{x}, \hat{r})]
\end{aligned} \tag{3.36}$$

As we can see, the gradient is an expectation with respect to the policy $P(\hat{x}|u,i)$. MRT is a method which directly follow the sentence probability and approximates the expectation by sampling $\hat{x}$. It regards each $\hat{x}$ as an action. However, the whole space of $\hat{x}$ is way too large. The sampling can hardly approximate, so results show very high variance.

Word-level MC search is another policy gradient method. It sees the each word prediction is an action and the feedback value is deferred till the end. So the generation becomes a sequence of actions with delayed value feedback. To estimate the intermediate action value, it applies MC search for each sampled word $\hat{w}_t$:

$$\begin{aligned}
\hat{X}_t &= \{\hat{x}_1, \hat{x}_2, ..., \hat{x}_n\} = MC_{u,i}(\hat{x}_{1:t-1}, \hat{w}_t) \\
\Delta(\hat{x}_{1:t-1}, \hat{w}_t, \hat{r}) &= \frac{1}{|\hat{X}_t|} \sum_{\hat{x}_k \in \hat{X}_t} \Delta(\hat{x}_k, \hat{r})
\end{aligned} \tag{3.37}$$

where $\hat{x}_{1:t-1}$ is the prefix sequence and $\hat{X}_t$ is the MC sampled set. $MC_{u,i}$ samples $n$ sentences, each of which continues with context $\hat{x}_{1:t-1}$ and input $\hat{w}_t$. The action value $\Delta(\hat{x}_{1:t-1}, \hat{w}_t, \hat{r})$ for $\hat{w}_t$ is the average of the MC sampled set. Then we change the objective in Equation 3.38 to minimizing the action value at each word. Similarly, we can have:

$$\begin{aligned}\nabla_\Theta L^{cr} &= \nabla_\Theta \sum_{u,i} \mathbb{E}_{P(\hat{w}_t|u,i,x_{1:t-1})}[\Delta(\hat{x}_{1:t-1}, \hat{w}_t, \hat{r})] \\ &= \sum_{u,i} \mathbb{E}_{P(\hat{w}_t|u,i,x_{1:t-1})}[\nabla_\Theta log P(\hat{w}_t|u, i, x_{1:t-1}) \cdot \Delta(\hat{x}_{1:t-1}, \hat{w}_t, \hat{r})]\end{aligned} \tag{3.38}$$

However, since it requires to conduct MC search at each position in a sequence to estimate the action value, the training process becomes extraordinary time-consuming, so we replaced the policy gradient with gumbel softmax in our final model.

### 3.3.3 Gumbel Softmax Sample

Gumbel Softmax Sample can estimate gradient for categorical sampling. Briefly speaking, it contains two parts: reparameterize the randomness in sampling to a gumbel distribution; simulate a relaxed one-hot vector with softmax. In our case, we need a strict one-hot vector to represent one word, so we employ the Straight-Through (ST) Gumbel Softmax estimator [Jang et al., 2016] which outputs a one-hot vector but uses the relaxation in back propagation to approximate the gradient. Then we can finally bridge the gap between predicted word distribution $y_t$ and the sentiment alignment loss $L^{cr}$:

$$o_t = ST\_GumbelSoftmaxSample(y_t) \tag{3.39}$$

where $o_t \in \mathbb{R}^{|V|}$ is the one-hot word vector whose only non-zero dimension indicates the index of the sampled word. It can also be easily mapped to its word embedding if necessary:

$$v_t = V o_t \tag{3.40}$$

The sampled sentence is re-defined as the sequence of the sampled one-hot vector $\hat{x} = \{o_1, o_2, ..., o_{|\hat{x}|}\}$ in Equation 3.34 so that $L^{cr}$ can be trained end-to-end.

### 3.3.4 Adversarial Training

Our architecture needs the adversarial training to regularize the readability in text generation to avoid feeding non-sense words into the content rater. The content discriminator $CD$ is added to play as the opponent of our multitask model, specifically the generator. It learns to differentiate if the explanation is authentic $x$ or generated $\hat{x}$. Our design does not limit the algorithms of the discriminator. In our implementation, it is made of a bidirectional RNN and an MLP binary classifier. The output of $CD$ is a probability representing how likely the input is positive. $CD$ is trained in cross-entropy loss with the ground truth explanations $x$ as positive and ones sampled from the generator as negative:

$$L^D = -\frac{1}{|X|}\sum_{x \in X} logCD(x) - \sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[log(1 - CD(\hat{x}))] \tag{3.41}$$

On the other hand, the objective of the generator is to trick the discriminator to treat the sampled explanation $\hat{x}$ as positive. So the adversarial loss of the generator is:

$$L^{cd} = -\sum_{u,i} \mathbb{E}_{P(\hat{x}|u,i)}[logCD(\hat{x})] \tag{3.42}$$

As we can see, the loss also requires sampled $\hat{x}$ as the input like Equation 3.34. Therefore, it takes use of the Gumbel Softmax sampling process as well for end-to-end back propagation.

### 3.3.5 Overall Objective

The overall objective of our model is formulated as:

$$J = \min_{\Theta}(\lambda_r L^r + \lambda_b L^b + \lambda_x L^x + \lambda_{cr} L^{cr} + \lambda_{cd} L^{cd} + \lambda_n ||\Theta||^2) \tag{3.43}$$

where $\Theta$ is all the model parameters and $\lambda$ are the loss weights.

## 3.4 Training

Due to our model's complex structure, the training is challenging to fully leverage its potential. The whole process can be split into five steps. First, train the independent content rater. It does not depend on anything in our model. Second, pre-train the rater till its optimal. This is essential to learn a good sentiment vector which will be used in the generator. Third, freeze the rater and train the generator alone with likelihood loss. The untrained generator will back propagate noise to hurt the rater through messing the sentiment representations at the beginning, so the rater is frozen from updates. Moreover, the text generation is a much more complicated case than rating prediction. Train it separately can help align their pace in the combined training. Fourth, turn on both the rater and generator to conduct combined training. This step allows the model to further sharp the sentiment representation from both tasks. At last, freeze the rater again, and train the generator with the supervisor. The discriminator and generator should be trained in turn. During the training of the generator, it is updated by both likelihood loss, sentiment alignment loss and adversarial loss.

# Chapter 4

# Experiments

In this chapter, we quantitatively evaluate both the personalized recommendations and explanation generation of our model. For recommendation, rating and ranking are both assessed. For explanation quality, three aspects are addressed: readability, feature personalization, and sentiment alignment. We conduct the experiments on processed Yelp open dataset. Our model is compared against many state-of-the-art recommendation baselines and neural explainable recommendation baselines. Improved performance in both tasks can be witnessed in the results. We also run a qualitative analysis to approve our gates design contribute to the improvement. In the following, we use Sentiment Aligned Explainable Recommendation System (SAER) to refer our proposal for clarity.

## 4.1 Data Preprocessing

To test our model's capability, we choose the popular Yelp open dataset which has been largely exploited in many recommendation works [Li et al., 2016, Wang et al., 2010, Tao et al., 2019]. Focusing on catering business, the data contains user profiles, items profiles (e.g. restaurants, hotels, casinos), and reviews, each of which includes a numerical rating in $[1, 2, 3, 4, 5]$ and a multi-sentence paragraph. It does not provide the features and explanations our solution needs. So at first, we use the Sentires toolkit [Zhang et al., 2014] to extract a raw list of popular aspects mentioned

| Reviews |
| --- |
| Off the grid Mexican in Vegas. **Very tasty, quality food and fantastic margaritas. Staff were very friendly as well.** This place will be a regular stop for me when In Vegas going forward. |
| Leave out the mussels. If they have the sea bass special, I highly recommend it. **The chicken parm and crab tortellini were also very good and very big.** |
| I have been coming here for years and this place is great. No, they did not try and rip me off on numerous occasions. |
| I loved the KC sauce because it had nice spicy kick to it. I didn't care much for the other 5 sauces but that is my own preference. |

Table 4.1: Example of unprocessed reviews. The bold ones can serve as recommendation explanations. Other text can not help readers to understand the item.

in reviews. Then we manually filter out some inappropriate terms based on domain knowledge. For example, the words "wife" and "mom" commonly appear in food reviews, but they are not features of restaurants. The left words form the feature list. While we may directly regard the original reviews as explanations, a large ratio of review text actually does not justify users' ratings [Ni et al., 2019]. Instead, shown in Table 4.1, many of them merely state experiences and subjective emotions. Generally, explanations should at least be descriptions of the item' features to help readers better understand it. To increase the truthfulness of our experiments, we decide to only use these feature descriptions as explanations. With the help of software like Stanza [Qi et al., 2020], we craft a basic rule to extract them: the sentence should contain at least one feature word which serves as the dependent in a "nsubj" relation of the grammatical dependency representation. Based on the crafted explanations, we pick the top $20,000$ words to build the vocabulary and map others to unknown. With all the sets ready, we run recursive filtering to get a subset of users and items with their corresponding ratings and explanations to alleviate the data sparsity issue. We keep each user to have 40 reviews minimum so that everyone can safely reserve 15 of them for ranking evaluation. The statistics of the processed data is summarized in Table

| # Users | # Items | # Reviews | % w/ Exp | # Features |
|---|---|---|---|---|
| 3,789 | 4,165 | 31,7854 | 79% | 509 |

Table 4.2: Statistics summary of processed data.

4.2. Note not all reviews have explanations due to the extraction rules previously introduced. We still decide to keep them because this better imitates the reality that fewer users are willing to write paragraphs of text than leaving a rating.

## 4.2 Baselines

To evaluate the personalized recommendation performance, we include the following baselines:

- **MF**: Matrix Factorization [Koren et al., 2009]. It is the classical collaborative filtering algorithms used in recommendation systems. It decomposes the rating matrix into the product of two lower dimensional matrices.

- **NMF**: Non-negative Matrix Factorization [Lee and Seung, 2001]. It is similar to MF, but the parameters are non-negative. It is also widely applied for recommendation.

- **PMF**: Probabilistic Matrix Factorization [Mnih and Salakhutdinov, 2008]. Latent factor model with Gaussian distribution.

- **SVD**: Singular Value Decomposition [Koren, 2008]. It utilizes implicit feedback information for latent factor modeling.

- **GMF**: General Matrix Factorization [He et al., 2017]. It is a neural network version of MF with additional learned dimension weights. It is introduced as a sub-module in NCF.

- **MLP**: Multi-Layer Perceptron [He et al., 2017]. A generic neural network structure serves as the foundation for others including ours. It is also introduced as a sub-module in NCF.

- **NCF**: Neural Collaborative Filtering [He et al., 2017]. It is a generic matrix factorization framework which integrates the GMF and MLP.

We also set up two neural explainable recommendation baselines to compare the explanation quality as well as the personalized recommendation:

- **HSS**: Hierarchical Sequence-to-Sequence Model [Chen et al., 2019]. It is a multi-task model of rating prediction and explanation generation. It employs hierarchical RNN to model multi-sentence generation. It designs a topical feature attention and auto-denoising mechanism to focus on personalized feature generation.

- **NRT**: Neural rating and tips generation [Li et al., 2017b]. It is also a multi-task model of rating regression and content generation. Similar to us, it takes the sentiment correlation between the two tasks into modeling via a one-hot rating vector. It originally targets tip (a short sentence summary) generation, but it can be smoothly migrated to our scenario via treating explanations as tips.

## 4.3 Personalized Recommendation

For personalized recommendation, we evaluate both the rating and ranking prediction. The rating is measured in Root Mean Square Error (RMSE). The ranking is measured in Normalized Discounted Accumulative Gain (NDCG), specifically NDCG@{3,5,10,15}. Both of them are widely used in recommendation evaluation.

The detailed results in Table 4.3 show that our model SAER performs better in all categories. The improvement is RMSE is more obvious than NDCG. This is expected because, besides the BPR regularization, we do not directly target on pairwise ranking. The whole cooperative learning with explanations is aiming to get a

| | RMSE | NDCG@3 | NDCG@5 | NDCG@10 | NDCG@15 |
|---|---|---|---|---|---|
| MF | 0.9773 | 0.3834 | 0.5143 | 0.7453 | 0.8947 |
| NMF | 1.0645 | 0.3721 | 0.5027 | 0.7317 | 0.8873 |
| PMF | 0.9954 | 0.3780 | 0.5083 | 0.7400 | 0.8916 |
| SVD | 0.9711 | 0.3867 | 0.5186 | 0.7480 | 0.8973 |
| GMF | 0.9749 | 0.3838 | 0.5144 | 0.7453 | 0.8952 |
| MLP | 0.9788 | 0.3858 | 0.5169 | 0.7463 | 0.8964 |
| NCF | 0.9870 | 0.3801 | 0.5113 | 0.7409 | 0.8927 |
| HSS | 1.0423 | 0.3532 | 0.4777 | 0.7064 | 0.8721 |
| NRT | 0.9764 | 0.3861 | 0.5170 | 0.7459 | 0.8963 |
| SAER | **0.9649** | **0.3882** | **0.5198** | **0.7494** | **0.8980** |

Table 4.3: Personalized recommendation evaluation in RMSE and NDCG

more expressive point-wise latent representation. It worth noting the difference among MLP, NRT, and SAER. Both NRT and our model's rating module are essentially an MLP. NRT uses a shared latent space and a word-frequency prediction task to leverage the textual information from reviews. The results indicate that textual information can slightly contribute to the recommendation. The further progress showed in SAER approves our sentiment vector and corresponding soft gates are able to better distill and exploit the textual data.

## 4.4 Explanation Quality

Since explainable recommendation is a rapidly changing research field, there is not a commonly agreed criterion to verify the explanation quality yet. We highlight three aspects in our experiments: readability, feature personalization, and sentiment alignment. Readability is inevitable in all natural language generation tasks. As natural language explanations, delivering fluent and meaningful text that people can understand is mandatory. Personalized features impact explanations' persuasiveness.

| | BLEU-1 | BLEU-2 | BLEU-4 |
|---|---|---|---|
| HSS | 11.41 | 3.76 | 1.12 |
| NRT | 17.34 | 6.04 | 1.21 |
| SAER (w/o Sup) | 18.07 | 6.89 | 1.23 |
| SAER (w/ Sup) | **18.11** | **7.09** | **1.28** |

Table 4.4: BLEU scores of decoded explanations

Describing what users truly care about can better assist their decisions. The importance of this aspect has also been mentioned in some explainable works [Ni et al., 2019, Chen et al., 2019]. Sentiment alignment is the focus of this project. Explanations must address the sentiments expressed in recommendation with details. Conflicted sentiments confuse and alienate users.

In this section, we use two versions of our model, SAER (w/o Sup) and SAER (w/ Sup), which indicate with and without the sentiment supervisor module. They can help understand our model in the following experiments. Both SAER and NRT are trained to generate a single sentence. Since most reviews in our dataset contain three sentences, we sample three sentences for each user and item pair as the decode explanation. HSS is designed to cater multi-sentence scenario. It takes use of beam search to decode.

We measure the readability in BLEU [Papineni et al., 2002] score which is a popular precision-based metric. The results are shown in Table 4.4. SAER (w/ Sup) achieves the best scores in all BLEU categories. SAER (w/o Sup) also exceeds the baselines, which reveals that the sentiment is able to benefit explanation generation. Note NRT uses the final predicted rating in the initial hidden state of the generator. Its gap towards us suggests our sentiment vector is more representative than the end rating and our model's soft gates can better exploit the sentimental information than RNN's initial state. The additional boost brought by the supervisor module demonstrates the effectiveness of adversarial training. It not only protects the readability from collapse but further improves it.

| | Precision | Recall | F1 |
|---|---|---|---|
| HSS | **0.2947** | 0.1195 | 0.1700 |
| NRT | 0.1206 | 0.1246 | 0.1225 |
| SAER (w/o Sup) | 0.2010 | 0.1543 | 0.1746 |
| SAER (w/ Sup) | 0.1918 | **0.1634** | **0.1765** |

Table 4.5: Feature precision, recall and F1 of decoded explanations

The personalized feature prediction is evaluated by precision, recall, and F1. For each pair of ground truth and generated explanation, we first extract the feature words and count the overlap. Then divide the overlap with generated and ground truth feature counts respectively to get the precision and recall. Table 4.5 shows the experiment results. Since NRT is originally for tip generation and does not have the concept of feature during modeling, its relatively low performance is reasonable. HSS achieves a significantly higher precision, but our model maintains a better balance between the precision and recall so it wins in the comprehensive F1 score. There are two factors leading to this result. First, HSS decodes in a greedy manner (beam search) while SAER conducts multinomial sampling. Greedy decoding tends to stick with the safest options. Plus the Yelp data is very biased towards generic features, such as "food" and "service" which apply to a large portion of reviews. Therefore, HSS is trapped to repeat common features while maintaining high precision. The randomness within sampling helps our model escape from the safe options and explore other features. We also argue that these accurate but generic features are not helpful for explanations, because intuitively people may write general terms themselves but prefer to read specific features. For example, "the food is good" is popular in Yelp data but showing them back to users is less supportive to their decision making than rare but specific descriptions, like "the salmon sushi is good". The second factor is the different feature pool. HSS uses the whole feature list while in our feature gate, we filter a subset for each item. This technique makes the feature prediction less challenging for our model, although the results prove it to be a valuable trick for

| | RMSE-GT | RMSE-PRED |
|---|---|---|
| HSS | 1.3727 | 0.9653 |
| NRT | 1.2281 | 0.7704 |
| SAER (w/o Sup) | 1.1573 | 0.6573 |
| SAER (w/ Sup) | **1.0931** | **0.5351** |

Table 4.6: Sentiment alignment evaluation of decoded explanations. RMSE-GT is the RMSE between explanation rating and ground truth rating. RMSE-PRED is the RMSE between explanation rating and predicted rating.

explanation quality.

Offline evaluation of sentiment alignment is tricky since it should be users to judge the text's sentiment. We use a separately trained text sentiment regression model to simulate a human judge. The model judge predicts a sentimental rating for the decoded explanations. We compute the RMSE between the explanation rating and the predicted rating. The value denotes the sentiment difference between the recommendation and explanations. The metric is marked as RMSE-PRED. We also compare the explanation rating with the ground truth rating in the same manner. The value is denoted as RMSE-GT. Table 4.6 presents the experiment results. Our model is significantly better than other baselines. The effect of the sentiment supervisor is very obvious. Even without it, SAER also outperforms others, which demonstrates the success of our sentiment modeling. It is surprising to notice that RMSE-GT is also improved as the RMSE-PRED improves. Especially considering SAER (w/ Sup) is only trained to match the predicted labels, the gap towards ground truth labels is reduced. One explanation is that the predicted ratings have a similar distribution as ground truth ratings through learning. So learning from predicted ratings may organize the explanation ratings into a closer distribution.

| Sample 1 | | | Sample 2 | | | Sample 3 | | |
|---|---|---|---|---|---|---|---|---|
| word | SG | FG | word | SG | FG | word | SG | FG |
| great | **0.35** | 0.08 | the | 0.13 | 0.06 | the | 0.16 | 0.02 |
| atmosphere | 0.18 | **0.65** | food | 0.15 | **0.58** | patio | 0.14 | **0.60** |
| & | 0.13 | 0.01 | here | 0.07 | 0.00 | is | 0.16 | 0.05 |
| food | 0.25 | **0.35** | is | 0.12 | 0.00 | a | **0.33** | 0.00 |
| is | 0.09 | 0.04 | consistently | **0.53** | 0.00 | great | **0.31** | 0.01 |
| delicious | **0.55** | 0.00 | tasty | **0.40** | 0.00 | choice | 0.11 | 0.10 |
| . | 0.11 | 0.00 | . | 0.10 | 0.00 | for | 0.03 | 0.00 |
| | | | | | | a | 0.06 | 0.18 |
| | | | | | | pizza | 0.15 | **0.32** |
| | | | | | | . | 0.04 | 0.12 |

Table 4.7: Samples of words and their values of sentiment and feature gates. SG is short for sentiment gate; FG is short for feature gate.

## 4.5 Qualitative Analysis

Since our model's sentiment modeling is based on the intuition of how human write, it is necessary to analyze if the sentiment gate and feature gate work as expected. Although previous experiments imply their effectiveness, we need to study what they really learned to understand why the improvement happens.

In every decoding step, the soft gate predicts a value between 0 and 1 to weight the incoming vector. The larger the value means the output will depend more on the incoming vector. Therefore, by checking the output word and the gate value, we can understand the relevance. Ideally, sentimental words should be decided by the recommendation sentiment and feature words should be decided by the user-item preference, so we expect to see large gate value for the corresponding words. Table 4.7 gives three examples. The values basically supports our intuition. For example, in sample 1, words "great" and "delicious" depend more on sentiments than others while the word "is" relates to neither sentiment nor features. Worth note in sample 3, the first "a" has the highest sentiment gate but the second one is quite low. Since the gate value is predicted by the RNN hidden state, this shows the model believes

sentiments should be expressed after word "is", which is actually reasonable.

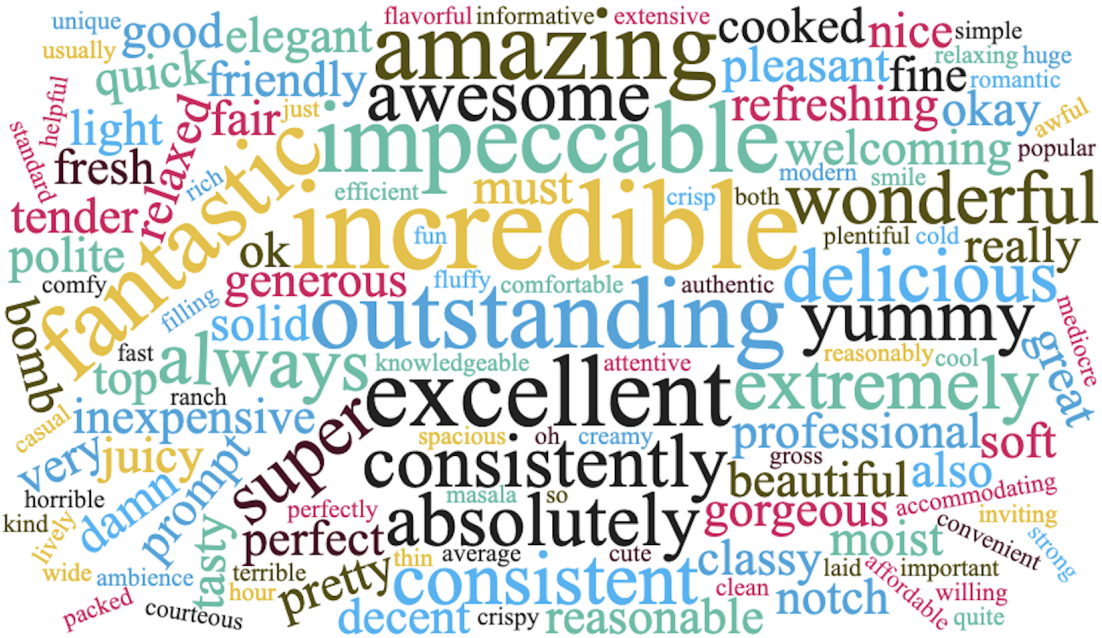


Figure 4-1: Heaviest dependents of the sentiment gate


Figure 4-2: Heaviest dependents of the feature gate

To understand the gates' overall contribution, two word clouds are given in Figure 4-1 and 4-2 to show their heaviest dependents in the whole generation corpus. At

first, we remove the low frequent words. Then for each word, its gate value is averaged across all its appearance. At last, we sort words in descending order according to their gate values and render the top ones in the word clouds where the font size reflects the average gate value. As a result, these two figures exactly agree with our design intuition. Our generator consults the sentiment to decide the strong opinionated words in Figure 4-1 and considers personal preference to mentions the aspects in Figure 4-2. It is safe to conclude our modeling is reasonable.

# Chapter 5

# Conclusion and Future Work

In this project, we present a new explainable recommendation model which can synthesize neural language explanation for recommendations. We claim it is critical for explanations to express consistent sentiments with recommendations. Motivated by this insight, our proposed solution generates sentiment aligned explanations to defend its recommendation decisions. Specifically, we design a neural architecture which binds rating prediction and text generation via latent sentiment modeling and connects their end objectives together through sentiment supervision. Experiments show our model not only improves in both recommendation and generation tasks over baselines, but also achieves significant progress in sentiment alignment.

This work initiates the exploration of sentiments' critical role in explainable recommendation. It leaves several valuable paths for our future works to pursue. The sentiment supervision in our current design, which trains the generated text to meet the predicted rating, does not involve any ground truth labels. This may lead to a semi-supervised learning where the models learn from each other. Considering the extreme sparsity of recommendation data, it may be able to exploit the dominant amount of unrated data to improve the performance. So far, sentiments are only considered in training but never in decoding stage, which still employs the basic greedy or multinomial sampling strategy. We may explore a new decoding strategy that selects words with the most promising sentiments. Monte Carlo search can be used to estimate the sentiment expectation at each decoding step. Moreover, recommendation is

eventually a group-wise ranking problem instead of point-wise rating problem. Therefore, it is vital to offer ranking supervision which guides the sentiments of multiple generated explanations to accurately reflect the relative order. At last, a large part of reviews cannot serve as explanations due to the writing style or format, but they also contain helpful information like sentiments and features. Introducing an intermediate knowledge graph into our pipeline can decouple the writing style and knowledge. While still exploiting qualified explanations to learn the generator, we can use the knowledge graph extracted from all reviews to refine the latent representations.

# Bibliography

[Almahairi et al., 2015] Almahairi, A., Kastner, K., Cho, K., and Courville, A. (2015). Learning distributed representations from reviews for collaborative filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems*, pages 147–154.

[Ayana et al., 2016] Ayana, S. S., Liu, Z., and Sun, M. (2016). Neural headline generation with minimum risk training. *arXiv preprint arXiv:1604.01904*.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bao et al., 2019] Bao, Y., Zhou, H., Huang, S., Li, L., Mou, L., Vechtomova, O., Dai, X., and Chen, J. (2019). Generating sentences from disentangled syntactic and semantic spaces. *arXiv preprint arXiv:1907.05789*.

[Bilgic and Mooney, 2005] Bilgic, M. and Mooney, R. J. (2005). Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI*, volume 5.

[Chen et al., 2019] Chen, H., Chen, X., Shi, S., and Zhang, Y. (2019). Generate natural language explanations for recommendation.

[Cho et al., 2014] Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

[Chung et al., 2014] Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

[Gers et al., 1999] Gers, F. A., Schmidhuber, J., and Cummins, F. (1999). Learning to forget: Continual prediction with lstm.

[Glynn, 1990] Glynn, P. W. (1990). Likelihood ratio gradient estimation for stochastic systems. *Communications of the ACM*, 33(10):75–84.

[Goodfellow et al., 2014] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

[Gunning, 2017] Gunning, D. (2017). Explainable artificial intelligence (xai). *Defense Advanced Research Projects Agency (DARPA), nd Web*, 2.

[He et al., 2015] He, X., Chen, T., Kan, M.-Y., and Chen, X. (2015). Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1661–1670.

[He et al., 2018] He, X., Du, X., Wang, X., Tian, F., Tang, J., and Chua, T.-S. (2018). Outer product-based neural collaborative filtering. *arXiv preprint arXiv:1808.03912*.

[He et al., 2017] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., and Chua, T.-S. (2017). Neural collaborative filtering. In *Proceedings of the 26th international conference on world wide web*, pages 173–182.

[Herlocker et al., 2000] Herlocker, J. L., Konstan, J. A., and Riedl, J. (2000). Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work*, pages 241–250. ACM.

[Herlocker et al., 2004] Herlocker, J. L., Konstan, J. A., Terveen, L. G., and Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, 22(1):5–53.

[Jang et al., 2016] Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

[Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[Kolda and Bader, 2009] Kolda, T. G. and Bader, B. W. (2009). Tensor decompositions and applications. *SIAM review*, 51(3):455–500.

[Koren, 2008] Koren, Y. (2008). 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.

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

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

[Li et al., 2016] Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., and Dolan, B. (2016). A persona-based neural conversation model. pages 994–1003.

[Li et al., 2017a] Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2017a). Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*.

[Li et al., 2019] Li, P., Wang, Z., Bing, L., and Lam, W. (2019). Persona-aware tips generation? In *The World Wide Web Conference*, pages 1006–1016.

[Li et al., 2017b] Li, P., Wang, Z., Ren, Z., Bing, L., and Lam, W. (2017b). 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.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81.

[Lu et al., 2018] Lu, Y., Dong, R., and Smyth, B. (2018). Why i like it: multi-task learning for recommendation and explanation. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 4–12.

[Lundberg and Lee, 2017] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in neural information processing systems*, pages 4765–4774.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Ma et al., 2008] Ma, H., Yang, H., Lyu, M. R., and King, I. (2008). Sorec: social recommendation using probabilistic matrix factorization. In *Proceedings of the 17th ACM conference on Information and knowledge management*, pages 931–940.

[McAuley and Leskovec, 2013] McAuley, J. and Leskovec, J. (2013). Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172.

[Mikolov et al., 2010] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. (2010). Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*.

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

[Murdoch et al., 2019] Murdoch, W. J., Singh, C., Kumbier, K., Abbasi-Asl, R., and Yu, B. (2019). Interpretable machine learning: definitions, methods, and applications. *arXiv preprint arXiv:1901.04592*.

[Ni et al., 2019] Ni, J., Li, J., and McAuley, J. (2019). 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.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). 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.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Qi et al., 2020] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., and Manning, C. D. (2020). Stanza: A Python natural language processing toolkit for many human languages.

[Ranzato et al., 2015] Ranzato, M., Chopra, S., Auli, M., and Zaremba, W. (2015). Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

[Rendle et al., 2012] Rendle, S., Freudenthaler, C., Gantner, Z., and Schmidt-Thieme, L. (2012). Bpr: Bayesian personalized ranking from implicit feedback. *arXiv preprint arXiv:1205.2618*.

[Ribeiro et al., 2016] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). " why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1135–1144.

[Rush et al., 2015] Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

[Schafer et al., 2007] Schafer, J. B., Frankowski, D., Herlocker, J., and Sen, S. (2007). Collaborative filtering recommender systems. In *The adaptive web*, pages 291–324. Springer.

[See et al., 2017] See, A., Liu, P. J., and Manning, C. D. (2017). Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*.

[Sharma and Cosley, 2013] Sharma, A. and Cosley, D. (2013). 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.

[Shen et al., 2015] Shen, S., Cheng, Y., He, Z., He, W., Wu, H., Sun, M., and Liu, Y. (2015). Minimum risk training for neural machine translation. *arXiv preprint arXiv:1512.02433*.

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

[Su and Khoshgoftaar, 2009] Su, X. and Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. *Advances in artificial intelligence*, 2009.

[Tao et al., 2019] Tao, Y., Jia, Y., Wang, N., and Wang, H. (2019). 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. ACM.

[Vinyals et al., 2015] Vinyals, O., Fortunato, M., and Jaitly, N. (2015). Pointer networks. In *Advances in neural information processing systems*, pages 2692–2700.

[Vinyals and Le, 2015] Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

[Wang et al., 2010] Wang, H., Lu, Y., and Zhai, C. (2010). Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 783–792.

[Wang et al., 2015] Wang, H., Wang, N., and Yeung, D.-Y. (2015). Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1235–1244.

[Wang et al., 2019] Wang, K., Hua, H., and Wan, X. (2019). Controllable unsupervised text attribute transfer via editing entangled latent representation. In *Advances in Neural Information Processing Systems*, pages 11034–11044.

[Wang et al., 2018] Wang, N., Wang, H., Jia, Y., and Yin, Y. (2018). 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.

[Williams, 1992] Williams, R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

[Yu et al., 2017] Yu, L., Zhang, W., Wang, J., and Yu, Y. (2017). Seqgan: Sequence generative adversarial nets with policy gradient. In *Thirty-First AAAI Conference on Artificial Intelligence*.

[Zeng et al., 2016] Zeng, W., Luo, W., Fidler, S., and Urtasun, R. (2016). Efficient summarization with read-again and copy mechanism. *arXiv preprint arXiv:1611.03382*.

[Zhang et al., 2019] Zhang, S., Yao, L., Sun, A., and Tay, Y. (2019). Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52(1):1–38.

[Zhang and Chen, 2018] Zhang, Y. and Chen, X. (2018). Explainable recommendation: A survey and new perspectives. *arXiv preprint arXiv:1804.11192*.

[Zhang et al., 2014] Zhang, Y., Zhang, H., Zhang, M., Liu, Y., and Ma, S. (2014). Do users rate or review? boost phrase-level sentiment labeling with review-level sentiment classification. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 1027–1030.