Why I like it: Multi-task Learning for Recommendation and Explanation

Yichao Lu
Layer 6 AI
yichao@layer6.ai

Ruihai Dong
Insight Centre for Data Analytics
University College Dublin
ruihai.dong@ucd.ie

Barry Smyth
Insight Centre for Data Analytics
University College Dublin
barry.smyth@ucd.ie

ABSTRACT

We describe a novel, multi-task recommendation model, which jointly learns to perform rating prediction and recommendation explanation by combining matrix factorization, for rating prediction, and adversarial sequence to sequence learning for explanation generation. The result is evaluated using real-world datasets to demonstrate improved rating prediction performance, compared to state-of-the-art alternatives, while producing effective, personalized explanations.

KEYWORDS

Recommender Systems; Natural Language Generation; Information Retrieval; Collaborative Filtering; Personalization

ACM Reference Format:
Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Why I like it: Multi-task Learning for Recommendation and Explanation. In *Twelfth ACM Conference on Recommender Systems (RecSys '18), October 2–7, 2018, Vancouver, BC, Canada.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3240323.3240365

1 INTRODUCTION

Recommender systems learn to predict the degree to which a user will like an item (*rating prediction*), which helps customers make better purchasing decisions and drives sales and loyalty for online businesses. Collaborative Filtering (CF) is a widely adopted recommendation approach, which predicts the interest of a user by collecting preference information, in the form of purchases or ratings, from many users; see [28, 40]. CF depends on user-item ratings, but the inherent sparsity of such ratings often limits accurate predictions. Modern CF algorithms attempt to exploit latent features [23] to represent users and items, which can lead to a lack of transparency for recommender systems. Such transparency issues can become severe when it comes to the case of e-commerce websites since latent features cannot be easily labelled. In order to build trust between a recommender system and its users, it has become important to complement recommendations with explanations so that users can understand why a particular item has been suggested [36, 49].

To improve the accuracy and explainability of recommender systems, the recommender systems community has recently tried to harness user-generated reviews as a new source of recommendation data [4, 9, 10, 12, 13, 18]. It is now feasible to automatically mine the opinions expressed by users in user-generated reviews. These mined opinions can supplement user preferences and product features, even when user-item ratings are scarce, and these opinions can often reveal hidden factors which influence customer choices [11, 14, 37]. Unsurprisingly, researchers have recently turned their attention to the potential of utilizing user-generated reviews as auxiliary information for enhancing recommendation performance [30]. Meanwhile, in an attempt to provide explainable recommendations, lately there has been a growing interest in the task of review generation. User-generated reviews can be viewed, in part at least, as explanations of the ratings given by users.

In this paper, we propose a novel multi-task learning framework that jointly learns to perform rating prediction and recommendation explanation from user-generated reviews. The traditional approach to recommendation explanation has an important limitation since it can only learn user preferences and item properties that are explicitly stated in observed reviews [44]. On the other hand, collaborative filtering techniques can discover the implicit interest of the user in the attributes of the item from customer feedback such as clicks or ratings. In the multi-task learning framework, the explanation generation process is augmented by the collaborative filtering model so that it can reveal the underlying factors that determine user's opinion towards an item even if they were never implied by the user. In addition, the multi-task learning framework encourages the model to provide an explanation together with its prediction. This effectively regularizes the model and enhances the generalization capability so that it can perform well on the unseen test dataset. Last but not least, jointly optimizing recommendation and explanation in a multi-task learning setting enforces consistency between the suggested recommendation and the provided explanation.

In what follows, we will demonstrate how we can integrate a sequence-to-sequence learning model with matrix factorization. Briefly speaking, we employ a matrix factorization model for rating prediction, and a sequence-to-sequence learning model for explanation generation, by generating personalized reviews for a given recommendation, user pair. We exploit the natural overlap between the latent factor vectors learned by matrix factorization and the textual features learned by the sequence autoencoder, allowing the individual models to regularize each other. The jointly trained model achieves strong results on both the task of rating prediction, beating the state-of-the-art, and on the task of recommendation explanation.

2 RELATED WORK

This paper attempts to extend recent progress on latent factor models and natural language processing [23, 31, 33, 39, 50], for review mining in a recommendation setting, by combining them into a multi-task learning framework. Latent factor models are commonplace in recommender systems research. They represent users and items as fixed-dimensional vectors encoding user preferences and item features. Matrix factorization based methods [23, 33, 39] are a popular class of latent factor models. By mapping both users and items to a joint latent factor space, matrix factorization can model user-item interactions as inner products of the user and item latent factor vectors. However, matrix factorization based methods are known to suffer from the aforementioned sparseness and transparency issues.

A proven and effective approach to solving the sparseness and transparency issues is to mine opinions and features from user-generated reviews. For example, the Hidden Factor as Topics (HFT) model [31] combines matrix factorization with Latent Dirichlet Allocation (LDA) [2] to utilize textual features from review texts. The authors demonstrate how the topic-level word clusters can help to interpret the "hidden factors" learned by matrix factorization. In [50], the authors propose the Explicit Factor Model, which utilizes phrase-level sentiment analysis for identifying "topics" that are correlated with user ratings. Deep Cooperative Neural Networks (DeepCoNN) [51] model user-item interactions based on review texts by utilizing a factorization machine model [38] on top of two convolutional neural networks [24]. The TransNets model [3] extends the DeepCoNN model by introducing an additional layer that is regularized by the latent representation of review texts.

Recently, sequence-to-sequence (seq2seq) deep neural network models [42] have been used in various tasks, ranging from text summarization [34] to question answering [52]. In particular, researchers have found that special seq2seq learning models called sequence autoencoders can be utilized in tasks such as sentiment analysis [6] and topic modeling [25]. These seq2seq generative models are also exploited to generate customer reviews for addressing recommendation explanation issues. For example, [45] proposed the opinion recommendation model which learns to generate customized review scores along with customized reviews.

Inspired by the success of Generative Adversarial Nets [16] in the computer vision literature, in recent years, the idea of adversarial training for natural language processing has been extensively explored. These adversarially trained models have demonstrated superior performance to their maximum likelihood counterparts, particularly in the task of natural language generation [27]. For example, the SeqGAN model [48] introduced a policy gradient based approach to effectively training generative adversarial nets for sequences generation tasks. Similarly, [26] utilizes adversarial training for open-domain dialogue generation. In order to address the *non-differentiability problem* associated with sequence generation frameworks, both models utilize the REINFORCE algorithm [46] together with Monte Carlo search for estimating the gradients.

3 A MULTI-TASK LEARNING APPROACH

In this section, we present the detail of our proposed recommendation model, which jointly learns to predict the rating of a target user for an item, and to generate a review, to serve as an explanation, for a particular user, item, or user-item pair. The overall architecture of the proposed model is presented in Figure 1.

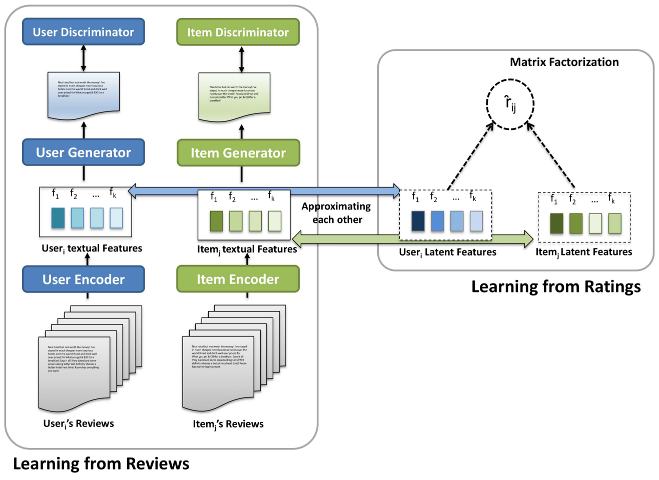

Figure 1: System architecture of multi-task learning model.

We begin by describing an adversarial seq2seq learning model that is capable of generating reviews for each user or item, which helps to improve the interpretability of latent factor models. Then we demonstrate how we can extend the base model to generate personalized customer reviews for specific items. After that, we introduce a context-aware extension of the probabilistic matrix factorization model that incorporates textual features learned from review documents. Finally, we propose a multi-task learning model that simultaneously learns to perform rating prediction and review generation, followed by a computational framework for optimizing the parameters.

3.1 Adversarial Sequence-to-Sequence Learning

We propose an encoder-decoder architecture capable of generating relevant reviews conditioned on observed review documents. To begin with, we need to define the concept of the *review document*, which is used throughout the paper. We refer to the set of all reviews written by user i as the *user review document* $d_{u,i}$. Similarly, the collection of reviews written for item j is referred to as the *item review document* $d_{v,j}$. Note that $d_{u,i}$ and $d_{v,j}$ may not be mutually exclusive, as a review written by user i on item j will be both included in $d_{u,i}$ and $d_{v,j}$. In other words, $d_{u,i} \cap d_{v,j} \neq \emptyset$ if and only if user i has written a review on item j.

We assume that user and item review documents are inherently different from each other. The former are indicative of user preferences, while the latter describe the features of items. For example, if "price" is frequently mentioned in a user's review document, we may infer that the price of the product would greatly affect the user's purchasing behavior. On the other hand, if a majority of reviews in an item's review document complain about the "poor quality", it is likely that the product suffers from serious quality issues. In order to capture such differences, we employ the same network architecture for modeling user and item review documents, but with different sets of parameters. For conciseness we only describe the user review document modeling framework in detail, noting that a similar approach is taken for item review documents but with item-specific details instead of user-specific ones.

The traditional approach to optimizing seq2seq models works by feeding ground-truth annotations at each time step during training. Such optimization strategy can lead to the problem of *exposure bias* [47]; i.e., at test time, the model is exposed to its own predictions so that the error accumulates through time. In order to combat this, we employ an adversarial training process, in which we simultaneously train a generator network and a discriminator network. The former learns to create fake reviews that can fool the discriminator, while the latter aims at detecting these adversarial samples.

3.1.1 Recurrent Review Generator. Given the user review document $d_{u,i}$, we expect the user review generator to generate reviews that are likely to be written by user i. To this end, we employ a recurrent neural network based, seq2seq learning framework consisting of two primary components: (1) a user document encoder and (2) a user review decoder. For user i, the user document encoder first maps the review document $d_{u,i}$ onto a D-dimensional vector $\widetilde{U_i}$, which can be thought of as the textual features extracted from the review document. Next, based on the textual feature vector $\widetilde{U_i}$, the user review decoder generates a review that is coherent with those in the review document. Both the user document encoder and the user review decoder are built using the Gated Recurrent Unit (GRU) [5]. The advantage of utilizing a GRU instead of the vanilla recurrent unit is that the internal gating mechanism of the GRU enables it to capture long-term dependencies in sequences.

Now we describe the architectural detail of the recurrent review generator. Suppose we have a user i and the corresponding review document $d_{u,i}$. For each review in the document, the user document encoder first maps each word w_t in the word sequence $(w_1, w_2, ..., w_T)$ into the respective k-dimensional vector representation $x_t \in \mathcal{R}^k$ through an embedding lookup operation. The embedding matrix is initialized with pre-trained word vectors obtained from *word2vec* [32], and then fine-tuned with back-propagation.

The word vectors are then fed into a bidirectional GRU which consists of a forward GRU that reads the input sequence in the usual order and a backward GRU that reads the input sequence in the reverse order. Utilizing a bidirectional GRU enables the document encoder to effectively summarize long sequences without loss of information. We concatenate the forward activation and backward activation at the last time step to obtain the vector representation of the review; i.e., $h_T = \left[\vec{h_T}, \overleftarrow{h_T}\right]$, where $\vec{h_T}$ and $\overleftarrow{h_T}$ are the hidden activations at the last time step for the forward GRU and the backward GRU, respectively.

After obtaining the vector representations for all the reviews in the review document $d_{u,i}$, the textual feature vector $\widetilde{U_i}$ for user i is then generated by averaging all the vectors. The textual feature vector is expected to encode all the necessary information about the user, based on which the user review decoder could generate user-specific reviews.

Given the textual feature vector $\widetilde{U_i}$, the user review decoder learns to generate user-specific reviews that reflect the encoded user preferences, through estimating the conditional probability

$$p(y_{i,1}, ..., y_{i,T'} \mid \widetilde{U_i}) = \prod_{t=1}^{T'} p(y_{i,t} \mid \widetilde{U_i}, y_{i,1}, ..., y_{i,t-1}), \qquad (1)$$

where $(y_{i,1}, ..., y_{i,T'})$ is the predicted review with length T'.

The user review decoder utilizes a single decoder GRU that iteratively generates reviews word by word. At time step t, the decoder GRU first embeds the output word $y_{i,t-1}$ at the previous time step into the corresponding word vector $x_{i,t-1} \in \mathcal{R}^k$, and then concatenates it with the user textual feature vector $\widetilde{U_i}$, i.e., $x'_{i,t} = \left[x_{i,t-1}, \widetilde{U_i}\right]$. The concatenated vector is provided as input into the decoder GRU to obtain the hidden activation h_t. Then the hidden activation is multiplied by an output projection matrix and passed through a softmax over all the words in the vocabulary to represent the probability of each word given the current context. The output word $y_{i,t}$ at time step t is sampled from the multinomial distribution given by the softmax. Note that, at each time step, the user textual feature vector $\widetilde{U_i}$ is fed into the decoder GRU along with the word vectors. This ensures that user-specific information is well maintained during the process of generating long sequences.

We assume that the output word $y_{i,0}$ at time step 0 is a special *SOS* token that indicates the start of the sequence, and that the hidden activation h_0 is a vector of all zeros. The review decoder iteratively generates word tokens until it outputs an *EOS* token, which stands for the end of the sequence.

3.1.2 Convolutional Review Discriminator. While the user review generator endeavors to generate reviews that are likely to be written by the users, the user review discriminator learns to distinguish the adversarial samples from authentic reviews. Given a candidate review and a target user, the user review discriminator utilizes two decision criteria to determine whether the review is indeed written by the user. Firstly, and most importantly, the discriminator determines the extent to which the given review seems to be written by human beings. If the review is written in a style that is rarely observed in human-generated texts, the discriminator can immediately make the decision with high confidence that the review is not an authentic one. Secondly, the discriminator determines the likelihood that the review is generated by the target user. This ensures that the user review generator cannot simply produce a human-readable text, but further constrains the review to be topically relevant to the observed reviews in the user review document.

The network architecture of the convolutional review discriminator is a slight variant of the convolutional neural net commonly utilized for document classification [21]; see Figure 2. To begin with, each word in the review is mapped to the corresponding word vector, which is then concatenated with a user-specific vector that identifies user information. The user-specific vectors are learned together with other parameters during training. The concatenated vector representations are then processed by a convolutional layer, followed by a max-pooling layer and a fully-connected projection layer. The final output unit is a sigmoid non-linearity, which squashes the probability into the [0, 1] interval.

3.1.3 Adversarial Training for Review Generation with REINFORCE. Now we describe the REINFORCE algorithm for the adversarial training of review generation. We assume that the generator G_θ is

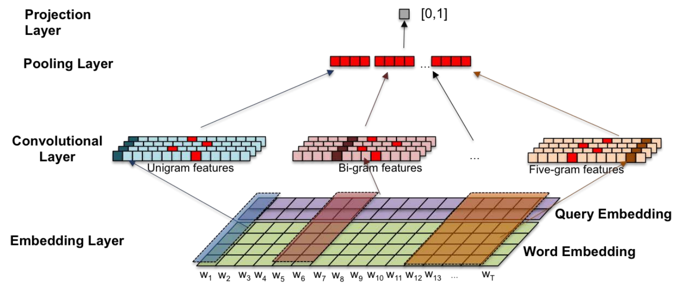

Figure 2: Convolutional Review Discriminator.

a virtual agent whose objective is to receive as much cumulative reward per trial as possible. The reinforcement (reward), on the other hand, is assigned by the discriminator D_ϕ based on its confidence that the review produced by the generator is an authentic one. In other words, the goal of the generator is to learn to create fake reviews that resemble ground-truth reviews so as to fool the discriminator with its generated samples.

The discriminator network in our model is trained via stochastic gradient ascent. The training objective of the discriminator is to minimize the probability of classifying adversarial samples to be authentic, while maximizing the probability of assigning correct labels to ground-truth reviews. The training objective can thus be expressed by the function as follows:

$$max_\phi \; E_{Y \sim p_{data}} \left[logD_\phi(Y) \right] + E_{Y' \sim G_\theta} \left[log(1 - D_\phi(Y')) \right], \quad (2)$$

where $Y \sim p_{data}$ refers to samples drawn from ground-truth reviews and $Y' \sim G_\theta$ refers to samples produced by the generator.

For each training step of the discriminator, we first sample a set of K ground-truth reviews $\{Y_i \mid Y_i \sim p_{data}, i = 1, \ldots, K\}$ and another set of K machine-generated reviews $\{Y'_i \mid Y'_i \sim G_\theta, i = 1, \ldots, K\}$. Then we compute the gradient of the discriminator with respect to the objective function, and employ gradient ascent to update the parameters of the discriminator.

Due to the fact that sequence generation is carried out through a discrete sampling process, the gradient cannot be directly back-propagated from the discriminator to the generator. In order to bypass the indifferentiability problem, we employ the policy gradient algorithm [46] to estimate the gradient of the objective function $J(\theta)$ with respect to the generator's parameters θ:

$$\begin{aligned} \nabla_\theta J(\theta) &= E_{Y' \sim G_\theta} \nabla_\theta logG_\theta(Y') \cdot [R(Y') - b(Y')] \\ &\approx \frac{1}{K} \sum_{i=1}^{K} \nabla_\theta logG_\theta(Y'_i) \cdot \left[R(Y'_i) - b(Y'_i) \right], \quad (3) \end{aligned}$$

where $R(Y'_i)$ is the reward assigned by the discriminator to the i-th generated sequence Y'_i, and $b(Y'_i)$ denotes the baseline value for reducing the variance.

As in [26], we utilize Monte Carlo search for approximating the state-action values. Briefly speaking, in order to estimate the reward for each intermediate step, starting from a partially decoded sequence, we repeat the process of sampling tokens until finishing for K times. Then we average the rewards assigned for these K sequences with a shared common prefix to obtain the expected reward.

For the purpose of multi-task learning, we regularize the textual features learned by the document encoder with the user coefficient matrix learned by matrix factorization. The rationale is that such a setting would enable the seq2seq model to exploit the user preferences inferred from the collaborative filtering approach, instead of merely relying on features learned from review documents. The loss function of the generator can thus be written as

$$L(\theta) = -J(\theta) + \lambda ||U - \widetilde{U}||_F, \quad (4)$$

where U is the user coefficient matrix learned by matrix factorization, which we describe in Section 3.3. $\widetilde{U}$ is the matrix of the user textual features learned by the user document encoder and λ is a weighting factor that controls the relative importance of the reconstruction loss and the regularization loss.

Up to now, we have described the network architectures of the user review generator and the user review discriminator, together with the adversarial training process for optimization. As we mentioned earlier, the item review generator and the item review discriminator employ the same architectures as their user-specific counterparts, but with different parameters. In order to generate a review for item j, the item review generator first utilizes the item document encoder to extract the item textual feature $\widetilde{V_j}$ from the item review document $d_{v,j}$, and then employs the item review decoder to generate the review on a token by token basis.

3.2 Generating Personalized Explanation

In Section 3.1, we demonstrate that the adversarial seq2seq model is capable of generating reviews for each user or item. These generated reviews can help to reveal user preferences and item attributes, in the sense that they essentially summarize the features that distinguish users and items from each other. However, for the purpose of this paper, we are more interested in generating personalized review predictions for each item, which provides recommendation explanation at a finer level of granularity.

Now we describe the approach to generating a personalized recommendation explanation for user i on item j. As before, we first extract user and item textual features $\widetilde{U_i}$ and $\widetilde{V_j}$ from review documents $d_{u,i}$ and $d_{v,j}$, utilizing the user and item document encoders. Then, instead of employing separate review decoders for generating user-specific and item-specific reviews, we utilize a single review decoder for generating reviews for the current user-item pair. At each time step, we concatenate $\widetilde{U_i}$ and $\widetilde{V_j}$ together with the embedding vector of the current word before feeding it into the review decoder. For the purpose of adversarial training, we also train a review discriminator along with the review generator. The aim of the review discriminator is to identify whether a given review is written by user i on item j.

3.3 Context-aware Matrix Factorization for Rating Prediction

We extend the Probabilistic Matrix Factorization (PMF) model [33] by using textual features extracted from review documents to serve as regularizers for the user and item latent factor vectors.

Given N users and M items, the matrix factorization model factorizes the user-item rating matrix $R \in \mathcal{R}^{N \times M}$ into the product of the user coefficient matrix $U^T \in \mathcal{R}^{N \times D}$ and the item factor matrix $V \in \mathcal{R}^{D \times M}$. Essentially, matrix factorization embeds users and items into a D-dimensional vector space where row vector u_i and column vector v_j correspond to the inferred latent factor vectors for user i and item j, respectively.

We assume that the observed ratings are corrupted by zero-mean additive Gaussian noise with noise variance σ^2. Therefore, the conditional distribution over the observed ratings can be formulated as in Equation 5, where $\mathcal{N}(x \mid \mu, \sigma^2)$ is the probability density function of the Gaussian distribution with mean μ and variance σ^2, and I_{ij} is an indicator function where $I_{ij} = 1$ if user i rated item j and $I_{ij} = 0$ otherwise.

$$\boxed{p(R \mid U, V, \sigma^2) = \prod_{i=1}^{N} \prod_{j=1}^{M} \left[\mathcal{N}(R_{ij} \mid u_i^T \cdot v_j, \sigma^2) \right]^{\boxed{I_{ij}}}} \tag{5}$$

In order to address the sparsity problem associated with the traditional MF model, we assume that the textual features extracted from review documents can serve as informative indicators of the latent features. Therefore, we define the prior distributions of user and item latent factor vectors as:

$$\boxed{p(U \mid \widetilde{U}, \sigma_U^2) = \prod_{i}^{N} \mathcal{N}(U_i \mid \widetilde{U}_i, \sigma_U^2 I),} \tag{6}$$

$$\boxed{p(V \mid \widetilde{V}, \sigma_V^2) = \prod_{j}^{M} \mathcal{N}(V_j \mid \widetilde{V}_j, \sigma_V^2 I),} \tag{7}$$

where $\widetilde{U}_i$ and $\widetilde{V}_j$ are the textual features extracted from the review documents of user i and item j, as in Section 3.1.

By introducing the textual features into the prior distributions of the latent variables, in this way, the context-aware matrix factorization model can effectively utilize the auxiliary information so as to address the sparsity problem related to traditional collaborative filtering approaches.

3.4 Optimization Methodology

So far we have described the seq2seq review generation model and the context-aware recommendation model. They use a shared knowledge structure to capture cross-domain similarity. An intuitive approach to optimizing the parameters is to define the overall loss function as the linear interpolation of the individual losses of the two models, and to update all the parameters with stochastic gradient descent until convergence.

Nevertheless, the joint training of the two models can be problematic. This is due to the fact that the latent factor vectors learned by MF are dependent on the textual features learned by the sequence autoencoder, and vice versa. In order for MF to yield great performance, we expect the sequence autoencoder to accurately model the textual features. Similarly, the sequence autoencoder relies on precisely learned latent factor vectors so as to generate coherent explanations. At the early stage of training, however, both the latent factor vectors and the textual features are inaccurate. This results in a dilemma for the training process, as the sequence to sequence review generation model and the context-aware recommendation model will be falsely guided by each other. Consequently, the stochastic gradient descent algorithm can easily become trapped in one of the undesired local minima.

Inspired by the expectation-maximization (EM) algorithm [35], we introduce an iterative optimization algorithm that alternates between the parameter update of the seq2seq review generation model and that of the context-aware recommendation model. Briefly speaking, when we are updating the parameters of one model, we temporarily fix the parameters of the other model. Such a training methodology helps to alleviate the dependency between models to facilitate training.

Suppose that the optimal $\widetilde{U}$ and $\widetilde{V}$ are known a priori, the posterior distributions over U and V can be formulated as

$$\begin{aligned} &\boxed{\max_{U,V} p(U, V \mid R, \widetilde{U}, \widetilde{V}, \sigma^2, \sigma_U^2, \sigma_V^2)} \\ &= \max_{U,V} p(R \mid U, V, \sigma^2) p(U, V \mid \widetilde{U}, \widetilde{V}, \sigma_U^2, \sigma_V^2), \end{aligned} \tag{8}$$

where

$$p(U, V \mid \widetilde{U}, \widetilde{V}, \sigma_U^2, \sigma_V^2) = p(U \mid \widetilde{U}, \sigma_U^2) p(V \mid \widetilde{V}, \sigma_V^2) \tag{9}$$

is the joint posterior distribution of U and V given $\widetilde{U}$, $\widetilde{V}$, σ_U^2, and σ_V^2.

Maximization of the posterior probability can be reformulated as minimization of its negative logarithm, which is given by

$$\begin{aligned} \mathcal{L}(U, V \mid R, \widetilde{U}, \widetilde{V}) = \frac{1}{2} \sum_{i}^{N} \sum_{j}^{M} I_{ij} (R_{ij} - U_i^T V_j)^2 \\ + \frac{\lambda_U}{2} \| U - \widetilde{U} \|_F^2 + \frac{\lambda_V}{2} \| V - \widetilde{V} \|_F^2, \end{aligned} \tag{10}$$

where $\lambda_U = \sigma^2 / \sigma_U^2$, and $\lambda_V = \sigma^2 / \sigma_V^2$.

As Equation 10 becomes a quadratic function with respect to U (or V) when V (or U) is treated as constant, the equation reaches its optimal solution when the gradient of U (or V) equals zero. Therefore we adopt the Alternating Least Squares (ALS) technique [19] which repeatedly optimizes one of U and V while temporarily fixing the other to be constant:

$$U_i = (V I_i V^T + \lambda_U I_K)^{-1} (V R_i + \lambda_U \widetilde{U}_i), \tag{11}$$

$$V_j = (U I_j U^T + \lambda_V I_K)^{-1} (U R_j + \lambda_V \widetilde{V}_j), \tag{12}$$

where $I_i \in \mathcal{R}^{M \times M}$ is a diagonal matrix with I_{ij} as its diagonal elements, and $R_i \in \mathcal{R}^M$ is a vector of R_{ij}. Recall that $I_{ij} = R_{ij} = 0$ if user i has not yet rated item j.

Conversely, when the latent factor vectors are known and fixed, we can use the algorithm described in Section 3.1 to iteratively update the generators and discriminators.

The full algorithm for optimizing the multi-task learning model is presented in Algorithm 1. To begin with, we randomly initialize all the variables in the model, including the user coefficient matrix $U \in R^{D \times N}$, the item factor matrix $V \in R^{D \times M}$, and the parameters in the user and item review generators and discriminators. Following [26], we employ teacher forcing as a supplement to the policy gradient. When utilizing teacher forcing, we feed ground-truth reviews to the generator for model updates.

Algorithm 1: Optimization Algorithm

```
for $epoch \leftarrow 1$ to $T$ do
    for $i \leftarrow 1$ to $N$ do
        Update $U_i$ with least square approximation:
            $U_i \leftarrow (VI_iV^T + \lambda_U I_K)^{-1}(VR_i + \lambda_U \widetilde{U_i})$
    end
    for $j \leftarrow 1$ to $M$ do
        Update $V_j$ with least square approximation:
            $V_j \leftarrow (UI_jU^T + \lambda_V I_K)^{-1}(UR_j + \lambda_V \widetilde{V_j})$
    end
    for $iteration \leftarrow 1$ to $T - step$ do
        Update $\theta_U$ and $\theta_V$ via teacher forcing.
    end
    for $iteration \leftarrow 1$ to $G - step$ do
        Update $\theta_U$ and $\theta_V$ via policy gradient.
    end
    for $iteration \leftarrow 1$ to $D - step$ do
        Update $\phi_U$ and $\phi_V$ via gradient ascent.
    end
end
```

4 EVALUATION

In this section we evaluate our approach on several large-scale, real-world datasets, comparing recommendation and explanation performance to a number of state-of-the-art, baseline approaches. Specifically, we use five publicly available datasets - including two datasets from the Yelp Dataset Challenge [1] and three others from Amazon - for the purpose of this analysis; see Table 1. The performance of each model is measured using 10-fold cross-validation.

Dataset	#users	#items	#ratings
Yelp 2013 (Y13)	1,631	1,633	78,966
Yelp 2014 (Y14)	4,818	4,194	231,163
Amazon Electronics (AE)	37,128	25,783	1,689,188
Amazon Video Games (AV)	24,303	10,672	231,780
Amazon Grocery (AG)	14,681	8,713	151,254

Table 1: Summary statistics for the evaluation datasets

4.1 Rating Prediction

To analyze recommendation performance, we utilize the Mean Squared Error (MSE) metric, which compares predicted ratings in the test set to the groundtruth ratings: $MSE = \frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r_i})^2$.

We examine the following baseline methods together with our multi-task learning model:

(i) **PMF**: Probabilistic Matrix Factorization (PMF) [33] is a factor-based model from a probabilistic point of view that performs well on very sparse and imbalanced datasets.
(ii) **HFT**: Hidden Factor as Topics (HFT) [31] is a novel recommendation technique that incorporates Latent Dirichlet Allocation [2] into matrix factorization. The model is able to simultaneously make recommendations and discover hidden topics.
(iii) **CTR**: Collaborative Topic Regression (CTR) [43] learns interpretable latent structure from user generated content so as to integrate probabilistic topic modeling into CF.
(iv) **JMARS**: Jointly Modeling Aspects, Ratings, and Sentiments (JMARS) [7] is a state-of-the-art recommendation model that is capable of uncovering aspect-level topics and sentiments.
(v) **ConvMF+**: Convolutional Matrix Factorization (ConvMF) [20] is a ConvNet [24] based model that addresses sparseness by mining attributes from item documents. ConvMF+ refers to the ConvMF model with pre-trained word embeddings.

The results for different models are presented in Table 2. We refer to our multi-task learning model as ***MT***, and ***MT-encoder*** and ***MT-decoder*** correspond to the model with encoder and decoder removed from the seq2seq model, respectively. For ***MT-encoder***, the model replaces the review document encoder with a simple embedding lookup operation. And for ***MT-decoder***, the model simply utilizes the encoder network to learn textual features from observed review documents. In both models, the training objective remains to be minimizing the difference between the textual features vectors learned by the explanation model and the latent factor vectors learned by matrix factorization.

As we can see the ***MT*** model outperforms all the baseline models by a wide margin and across all datasets. Moreover, all the differences are statistically significant at the level of $p < 0.05$ using a paired t-test [8]. When the encoder or decoder is removed from the model, the MSE increases, showing that both the encoder and the decoder help to improve rating prediction.

Model	Y13	Y14	AE	AV	AG
PMF	0.985	1.053	1.311	1.297	1.251
HFT	0.977	1.029	1.159	1.152	1.121
CTR	0.975	1.013	1.184	1.147	1.139
JMARS	0.970	0.998	1.128	1.133	1.114
ConvMF+	0.917	0.954	1.117	1.092	1.084
MT - encoder	0.874	0.922	1.071	1.065	1.037
MT - decoder	0.871	0.926	1.074	1.069	1.038
MT	**0.861**	**0.896**	**1.018**	**1.026**	**1.004**

Table 2: Recommendation performance in terms of MSE

4.2 Explanation Quality

We demonstrate superior rating prediction performance for our ***MT*** learner, but what about explanation quality? We plan to conduct a live-user trial as the ultimate test of recommendation effectiveness and explanation quality, but for now we continue with our off-line evaluation approach, estimating the quality of explanations using the common *perplexity* metric [1] for evaluating the goodness of language models and the *tf-idf* similarity metric for evaluating the relevance of the generated reviews compared with the ground-truth ones. These are not ideal metrics of explanation quality - hence the need for extrinsic, live-user evaluations in due course - but for now they provide a useful starting point.

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

The perplexity metric is defined as the exponent of the average negative log-likelihood per word. Lower perplexity implies higher log likelihood, and thus indicates a better language model. Alternatively, perplexity can be thought of as a "branching" factor, i.e., if we pick the word from the probability distribution given by the language model, how many times in average do we need to pick in order to get the right word. Of course the perplexity metric is not a typical way to evaluate explanation quality. In the absence of more direct live-user feedback it is used here as a proxy to show that explanations are being generated using a better language model (lower perplexity).

For the *tf-idf* similarity metric, each word in the review is first transformed using the "term frequency inverse document frequency" algorithm [41]: $\text{tf-idf}_{t,d} = (1 + \log f_{t,d}) \cdot \log(1 + \frac{N}{n_t})$, where $f_{t,d}$ is the "term frequency" of the word t in review d, and $\text{idf}(t) = \log \frac{N}{n_t}$ is the "inverse document frequency" of the word t in the entire review document. Then each review is projected into a vector whose length equals the number of vocabulary in the dataset. Each dimension of the vector corresponds to the tf-idf value of a particular word. To measure the relevance of two reviews, we compute the cosine similarity between their vectors. Apparently, higher cosine similarity indicates a higher relevance of the reviews.

In brief, we generate explanations for users (***MT-U***), items (***MT-I***) and user-item pairs (***MT-P***). ***MT-U*** and ***MT-I*** refers to the user and item review generator described in Section 3.1, respectively, while ***MT-P*** corresponds to the personalized review generation model described in Section 3.2. Once again, we compare our approach to five state-of-the-art baselines:

(i) **N-gram**: The N-gram language model [22] is a classic model in the literature of statistical natural language processing, which models the probability of occurrence of N consecutive words. In this paper, we utilize the linear interpolation of unigram, bigram, and trigram for smoothing the probability distribution.
(ii) **Skip-gram**: The skip-gram model [32] is a log-linear language model that predicts the probability of next word given the context.
(iii) **LSTM**: LSTM refers to the recurrent language model that employs the Long Short Term Memory (LSTM) [17] as the recurrent cell.
(iv) **Opinosis**: Opinosis [15] is a graph based abstractive summarization framework that is capable of eliminating redundant opinions. As Opinosis is not essentially a language model, we only evaluated its performance in terms of *tf-idf* similarity.
(v) **GCN**: Generative Concatenative Network (GCN) [29] extends the recurrent language model by concatenating user- and item-specific information with the word vectors. GCN represents each user/item with a unique vector representation. At each time step, the user/item latent vectors are fed into the network together with the word embedding of the input token, thus ensuring long-range consistency in the generation process.

The results are presented in Table 3 and Table 4. Again, we see benefits accruing to our approach. The personalized ***MT-P*** approach tends to perform best of all of the ***MT*** approaches across all of the test datasets and in each case the ***MT*** results consistently outperform all of the baselines. Once again all differences are statistically significant at $p < 0.05$ level.

Model	Y13	Y14	AE	AV	AG
N-gram	89.3	82.1	68.5	73.7	79.6
Skip-gram	85.2	79.8	65.2	67.2	74.2
LSTM	79.7	71.5	64.3	65.7	70.8
GCN-U	73.3	63.6	57.7	64.1	65.3
GCN-I	69.3	61.3	54.8	59.8	63.2
GCN-P	67.9	60.7	52.9	57.6	62.1
MT-U	66.1	62.5	53.6	59.2	60.8
MT-I	65.6	60.2	52.8	54.1	58.3
MT-P	**65.2**	**59.4**	**51.4**	**53.7**	**57.6**

Table 3: Explanation performance in terms of perplexity

Model	Y13	Y14	AE	AV	AG
N-gram	0.007	0.009	0.005	0.009	0.011
Skip-gram	0.009	0.014	0.007	0.011	0.015
LSTM	0.019	0.022	0.014	0.017	0.019
Opinosis	0.029	0.031	0.029	0.025	0.027
GCN-U	0.042	0.040	0.039	0.041	0.038
GCN-I	0.048	0.044	0.045	0.045	0.046
GCN-P	0.051	0.047	0.045	0.047	0.046
MT-U	0.048	0.043	0.042	0.044	0.047
MT-I	0.051	0.045	0.046	0.049	0.049
MT-P	**0.053**	**0.052**	**0.049**	**0.052**	**0.051**

Table 4: Explanation performance in terms of tf-idf

4.3 Performance of the Discriminator

We carry out two experiments to evaluate the performance of the discriminator during the adversarial training process. The first experiment is designed to show the learning curve of the discriminator. After training the full multi-task learning model, we utilize the fully trained generator to create a set of adversarial samples. Then we repeat the training process from scratch, and monitor the accuracy of the discriminator at each time step. The experimental results on three datasets are presented in Figure 3. As we can see, at the early stage of training, a well-trained generator can easily fool the discriminator with its adversarial samples. However, through the training process, the discriminator gradually learns the key factors that help to distinguish adversarial samples from authentic ones.

Then we examine the extent to which a review discriminator can distinguish the reviews written by a user (or an item) from reviews of other users (or items). We carry out the experiment by shuffling half of the labels of user (or item) for the reviews in the test set, and let the discriminator predict the probability that the accompanied label of each review is its original one. As the discriminator has never observed these test reviews during training, and all the reviews are human-written texts, in such circumstances, the discriminator can only rely on the learned user and item feature vectors to distinguish corrupted samples. However, we find that the discriminator can do a great job. On the Amazon Electronics dataset, the user discriminator can make predictions at an accuracy of 79.4%,

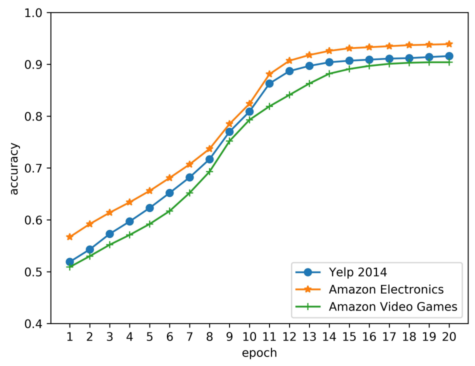

Figure 3: Learning curve of the discriminator network

while the item discriminator can be correct with a probability of 85.8%. The item discriminator has a better ability to do its job because the item review document is less flexible than the user review document. In both cases, it is obvious that the discriminator has indeed learned the important features of users and items. As otherwise, the accuracy should be around 50%.

4.4 Discussion

So far we have described a multi-task approach to recommendations, by co-learning rating prediction and explanation generation. In our off-line evaluation, we provide evidence to support the hypothesis that this approach can offer statistically significant improvements over state-of-the-art baselines, and that the multi-task nature of the approach improves both rating prediction and explanation generation. The results are consistent across a variety of common evaluation datasets but we do recognize that it will be appropriate to demonstrate that these benefits continue to be available in the context of real-life usage. This will require a live-user study, which remains a key objective for the future in this work.

Another weakness of the evaluation concerns the appropriateness of the perplexity measure as an arbiter of explanation quality. Although perplexity is commonly used as an intrinsic measure of language model goodness, it is not sufficient on its own to evaluate explanation quality, hence the need for user trials. In advance of this, however, it is also useful to examine some of the types of explanations that are produced in practice. To explore this we compared the predicted reviews with the ground-truth ones. We find that our model frequently learns to generate personalized reviews with a reasonable level of accuracy, although a detailed analysis of this requires a user-study and such is left as a matter for future work. For now we will instead provide some supporting examples of the types of explanations that are generated and their corresponding ground-truths.

For example, the review that our model predicts a user would write on a Nook eBook Reader is that "*Good reader. Comes with a screen protector. I am very pleased with my Nook. It is very comfortable to use - battery life is awesome. Screen works well under the sun.*" On the other hand, the ground-truth review is "*I loved my Nook very much, Its very comfortable to use. What I love about Nook more than kindle, that I could pick any Russian book, magazine on Internet download on my computer, convert to PD format and download on Nook + I could pick on Nook any size of letters for comfortable reading... On Kindle you can make bigger size of letters only if you Bought ebook on Kindle store. Other PD format books have so small size of letters, only if you zoom text, which I find uncomfortable. Like I said with Nook no problem. Also touch screen work great too.*" The user gives this item a rating of five stars.

It turns out too that the model has indeed discovered some insights into the recommendation. Even though the model was not explicitly told that the item was a Nook eBook reader, it inferred this from existing review documents. Also, the model discovered some important factors that were mentioned in the ground-truth review, e.g., comfort, screen, and the overall positive sentiment.

From this example we can see that the review generator works as expected: the generated review/explanation overlaps with the ground-truth to a large extent. However, in some cases, the review generator may provide convincing explanations even if the predicted review seems to be far from the ground-truth. For example, the generated review for another user is, "*Love my eBook. It has access to the web and can install apps.*", while the ground-truth is, "*This is a good reader. Uploads books fast. I do not do wifi on it because I have heard bad things about the updates. Also has problems if hooked up to a computer will take the shelves away. I have lots of books so this is a big bad thing for me. But still is good.*" Apparently, the generated review is less similar to the ground-truth but this can be understood because the user talked about a personal experience that cannot be inferred from the past reviews, nor learned from the item review document. We cannot judge the generator to have failed merely as the generated review is dissimilar to the real one.

All of this points to the need for further evaluation, particularly for the explanation generation process. As discussed we plan to deal with this as part of a future live user trial in which recommendations and explanations can be evaluated in the context of a genuine recommendation setting.

5 CONCLUSIONS

In this paper we present a novel multi-task learning framework that simultaneously learns to perform rating prediction and recommendation explanation. We demonstrate how improved performance can be achieved by the joint training of the two tasks. Ratings prediction improves across all test datasets, compared to the current state-of-the-art. The explanations generated are also more similar to the ground-truth than when more conventional approaches are used. The logical next step for this work is to move from off-line evaluations to online, live-user studies, especially when it comes to evaluating explanation quality. In future work, we will invite human judges to qualitatively evaluate the naturalness and convincingness of the generated explanations, and to perform online user study and A/B testing to further validate the effectiveness of the proposed model when providing explanations.

ACKNOWLEDGMENTS

This work is supported by Science Foundation Ireland under Grant Number SFI/12/RC/2289.

REFERENCES

[1] Leif Azzopardi, Mark Girolami, and Keith Van Risjbergen. 2003. Investigating the relationship between language model perplexity and IR precision-recall measures. In *International ACM SIGIR Conference on Research and Development in Informaion Retrieval*. 369–370.

[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.

[3] Rose Catherine and William Cohen. 2017. Transnets: Learning to transform for recommendation. In *Proceedings of the Eleventh ACM Conference on Recommender Systems*. ACM, 288–296.

[4] Li Chen, Guanliang Chen, and Feng Wang. 2015. Recommender systems based on user reviews: the state of the art. *User Modeling and User-Adapted Interaction* 25, 2 (2015), 99–154.

[5] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio. 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. *ArXiv e-prints* (Dec. 2014). arXiv:1412.3555

[6] Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. 3079–3087.

[7] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (jmars). In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 193–202.

[8] Thomas G Dietterich. 1998. Approximate statistical tests for comparing supervised classification learning algorithms. *Neural computation* 10, 7 (1998), 1895–1923.

[9] Ruihai Dong, Michael P O'Mahony, Markus Schaal, Kevin McCarthy, and Barry Smyth. 2013. Sentimental product recommendation. In *Proceedings of the 7th ACM conference on Recommender systems*. ACM, 411–414.

[10] Ruihai Dong, Michael P O'Mahony, and Barry Smyth. 2014. Further experiments in opinionated product recommendation. In *International Conference on Case-Based Reasoning*. Springer, 110–124.

[11] Ruihai Dong, Markus Schaal, Michael P O'Mahony, Kevin McCarthy, and Barry Smyth. 2013. Mining features and sentiment from review experiences. In *International Conference on Case-Based Reasoning*. Springer, 59–73.

[12] Ruihai Dong, Markus Schaal, Michael P O'Mahony, Kevin McCarthy, and Barry Smyth. 2013. Opinionated product recommendation. In *International Conference on Case-Based Reasoning*. Springer, 44–58.

[13] Ruihai Dong, Markus Schaal, Michael P O'Mahony, and Barry Smyth. 2013. Topic Extraction from Online Reviews for Classification and Recommendation.. In *IJCAI*, Vol. 13. 1310–1316.

[14] Ruihai Dong and Barry Smyth. 2017. User-based opinion-based recommendation. In *Twenty-Sixth International Joint Conference on Artificial Intelligence (IJCAI-17), Melbourne, Australia, August 19-25 2017*. International Joint Conferences on Artificial Intelligence.

[15] Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics*. Association for Computational Linguistics, 340–348.

[16] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *International Conference on Neural Information Processing Systems*. 2672–2680.

[17] Alex Graves. 2012. Supervised sequence labelling. In *Supervised sequence labelling with recurrent neural networks*. Springer, 5–13.

[18] John Hannon, Mike Bennett, and Barry Smyth. 2010. Recommending twitter users to follow using content and collaborative filtering approaches. In *Proceedings of the fourth ACM conference on Recommender systems*. ACM, 199–206.

[19] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on*. Ieee, 263–272.

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

[21] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[22] Reinhard Kneser and Hermann Ney. 1995. Improved backing-off for m-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, Vol. 1. IEEE, 181–184.

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

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

[25] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015. A hierarchical neural autoencoder for paragraphs and documents. *arXiv preprint arXiv:1506.01057* (2015).

[26] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547* (2017).

[27] Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *Advances in Neural Information Processing Systems*. 3155–3165.

[28] Greg Linden, Brent Smith, and Jeremy York. 2003. Amazon.com recommendations: Item-to-item collaborative filtering. *IEEE Internet computing* 7, 1 (2003), 76–80.

[29] Zachary C Lipton, Sharad Vikram, and Julian McAuley. 2015. Generative Concatenative Nets Jointly Learn to Write and Classify Reviews. *arXiv preprint arXiv:1511.03683* (2015).

[30] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook*. Springer, 73–105.

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

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

[33] Andriy Mnih and Ruslan R Salakhutdinov. 2008. Probabilistic matrix factorization. In *Advances in neural information processing systems*. 1257–1264.

[34] Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023* (2016).

[35] Radford M Neal and Geoffrey E Hinton. 1998. A view of the EM algorithm that justifies incremental, sparse, and other variants. In *Learning in graphical models*. Springer, 355–368.

[36] John O'Donovan and Barry Smyth. 2005. Trust in recommender systems. In *Proceedings of the 10th international conference on Intelligent user interfaces*. ACM, 167–174.

[37] Michael P O'Mahony and Barry Smyth. 2018. From opinions to recommendations. In *Social Information Access*. Springer, 480–509.

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

[39] Ruslan Salakhutdinov and Andriy Mnih. 2008. Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. In *Proceedings of the 25th international conference on Machine learning*. ACM, 880–887.

[40] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning*. ACM, 791–798.

[41] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[42] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *International Conference on Neural Information Processing Systems*. 3104–3112.

[43] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 448–456.

[44] Weiquan Wang and Izak Benbasat. 2007. Recommendation agents for electronic commerce: Effects of explanation facilities on trusting beliefs. *Journal of Management Information Systems* 23, 4 (2007), 217–246.

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

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

[47] Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* (2016).

[48] Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. SeqGAN: Sequence Generative Adversarial Nets with Policy Gradient. In *AAAI*. 2852–2858.

[49] Yongfeng Zhang and Xu Chen. 2018. Explainable Recommendation: A Survey and New Perspectives. *arXiv preprint arXiv:1804.11192* (2018).

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

[51] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 425–434.

[52] Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer sequence learning with neural networks for answer selection in community question answering. *arXiv preprint arXiv:1506.06490* (2015).