Estimating Reactions and Recommending Products with Generative Models of Reviews

Jianmo Ni and **Zachary C. Lipton** and **Sharad Vikram** and **Julian McAuley**
Department of Computer Science and Engineering
University of California San Diego
`jin018, zlipton, svikram, jmcauley@ucsd.edu`

Abstract

Traditional approaches to recommendation focus on learning from large volumes of historical feedback to estimate simple numerical quantities (Will a user click on a product? Make a purchase? etc.). Natural language approaches that model information like product reviews have proved to be incredibly useful in improving the performance of such methods, as reviews provide valuable auxiliary information that can be used to better estimate latent user preferences and item properties.

In this paper, rather than using reviews as an *inputs* to a recommender system, we focus on generating reviews as the model's *output*. This requires us to efficiently model text (at the character level) to capture the preferences of the user, the properties of the item being consumed, and the interaction between them (i.e., the user's preference). We show that this can model can be used to (a) generate plausible reviews and estimate nuanced reactions; (b) provide personalized rankings of existing reviews; and (c) recommend existing products more effectively.

1 Introduction

Review text has been extensively studied in modern recommender systems as a means of improving the performance on traditional recommendation tasks. Compared with conventional techniques that model simple numerical feedback (ratings, clicks, purchases, etc.), review text provides valuable information about user and item attributes, and more importantly the interaction between them. Recent systems have adapted ideas from topic modeling and sentiment analysis to leverage the side-information contained in reviews; in essence, these approaches use language models as a form of 'regularization,' such that the model should explain user preferences and review text simultaneously (McAuley and Leskovec, 2013; Bao et al., 2014; Ling et al., 2014).

In parallel, recent advances in generative text modeling have demonstrated the effectiveness of recurrent neural networks in capturing content, structure, and style in natural language. As a result, several recent works have focused on learning generative models of product reviews, either to generate reviews *per se*, or as a means of learning user and item attributes (Lipton et al., 2015; Radford et al., 2017; Dong et al., 2017; Hu et al., 2017). In this paper, we address the problem of how to leverage both review text and implicit feedback simultaneously in order to provide richer user experiences and more meaningful recommendations. In particular, we focus on estimating latent user preferences through implicit feedback while simultaneously predicting the contents of the reviews themselves. Thus, given an item that a user hasn't yet interacted with, our goals are to (a) generate a plausible review, in order to estimate the user's nuanced reaction to the product; (b) estimate whether the user would be likely to interact with the product, based on their learned latent attributes; and (c) rank existing reviews using the language model, in order to surface 'meaningful' reviews to the user.

To solve the above problems, we propose a model—CF-GGN—that combines Collaborative Filtering (CF) with Generative Concatenative Networks (GCN) (Lipton et al., 2015) to simultaneously perform item recommendation and review generation. Given a large dataset of implicit feedback (e.g. purchases vs. non-purchases), we start by modeling user and item latent factors through collaborative filtering; we also adapt ideas from text embeddings (e.g. word2vec), and use multi-layer perceptrons (MLP) to model complex interactions between embeddings and latent preference factors. Finally we build a generative concatenative model by stacking two LSTM layers; we adopt a simple replication strategy to concatenate latent factors with text input and jointly train the model in a supervised way. By sharing the same model parameters, the two tasks (recommendation and generation) mutually reinforce each other when all parameters are trained end-to-end.

To our knowledge, our work is the first to show that we can generate plausible reviews while simultaneously achieving substantial quantitative improvements on recommendation tasks. To summarize, our main contributions are as follows:

- We jointly perform recommendation and review generation by combining collaborative filtering with LSTM-based generative models. Our model captures latent information such as user preferences and item attributes, and learns to generate coherent, structured, and personalized reviews.

- We investigate the effect of sparsity in both reviews and implicit feedback data; a virtue of our joint training approach is that we can learn effective text models even for users with limited reviews at training time, i.e., we can learn to estimate likely reactions (in the form of reviews), on the basis of implicit feedback (e.g. purchases and clicks), even for users who write few reviews.

- We conduct extensive experiments on three real-word datasets in order to qualitatively and quantitatively demonstrate the effectiveness of our joint training approach.

2 Related Work

Review text provides valuable information about users' experiences and preferences toward the items they consume. Modeling the detailed information in reviews is especially important in sparse datasets, where a small amount of reviews carries substantially more information than is available in ratings alone. Several previous works have taken reviews into consideration to improve the performance of various recommendation tasks (McAuley and Leskovec, 2013; Ling et al., 2014; Bao et al., 2014; Wang et al., 2015).

Recently, neural-network-based models have been considered when modeling review text, in place of traditional topic modeling techniques. Zheng et al. (2017) leveraged Convolutional Neural Networks (CNN) to extract embeddings from reviews, which were used as features in a factorization machine (Rendle, 2010) to generate rating predictions. Catherine and Cohen (2017) proposed a model ('transNet') consisting of a source network and a target network; the former learns user and item embeddings from reviews while the latter performs sentiment analysis over the ground truth review. By minimizing the difference between the two networks' latent representation layers, the model learns more expressive latent factors. Wu et al. (2017) presented a neural network based model that jointly considers both ratings and reviews. Their approach models the temporal rating prediction task via a Recurrent Neural Network (RNN), and uses an additional LSTM network to model the review text as a regularization term in the loss function. Their experimental results show improvements on rating prediction tasks over state-of-the-art techniques, however they do not report results in terms of the model's ability to generate plausible reviews.

Given their expressive power when modeling sequential data, RNN-based methods have been widely studied for a variety of generation tasks (Graves, 2013; Zhang and LeCun, 2015; Sutskever et al., 2011). Recently, several works have focused on the task of learning language models to generate reviews. Generating coherent and personalized reviews still poses considerable challenges, due to their length, structure, and the sparsity of real datasets. Radford et al. (2017) trained a character-RNN language model based on the *Amazon* review dataset (McAuley et al., 2015); they consider a single multiplicative LSTM layer with 4096 hidden units and train the model for nearly a month. The authors find a sentiment unit among the hidden units which is highly correlated to review sentiment. They can then generate reviews by forcing this sentiment unit to be positive or negative. Lipton et al. (2015) proposed a generative concatenative network to supervise the training of a character-level language model; they concatenate auxiliary information in the form of one-hot representations of user/item IDs, categories and ratings. The model generates plausible reviews when conditioned on this auxiliary information, and demonstrates the potential for RNNs to capture subjective user information and generate personalized text, though they do not consider recommendation problems. Dong et al. (2017) studied attribute-to-sequence generative models with stacked LSTMs; they encode attributes such as user/item IDs and ratings and use the encoded information to initialize the hidden states of the LSTM layer. They also make use of an attention mechanism in the decoder to improve the accuracy of predictions. Their model is based on word-level LSTMs and struggles when generating long sequences. Moreover, their network does not have the ability to predict user preferences toward items, which limits performance when given unseen (user,item) pairs.

The most relevant work to ours is perhaps Li et al. (2017). They addressed the problem of jointly learning rating prediction and 'tip generation.' They employ a standard latent factor model with MLP layers to predict ratings, following which they use the latent factors to initialize the LSTM model for tip generation. Experimental results show quantitative improvements at rating prediction and their word-level generative model produces tips that are consistent with users' ratings. Compared with their task, we focus on review generation (rather than generating short tips), which requires the model to maintain generation quality over long sequences. Moreover, we consider review text as content during the training of our collaborative filtering model, which further enhances the prediction of user preferences.

3 Proposed Method

First we define our problem before introducing our CF-GCN model. Table 1 summarizes the notation used throughout this paper.

Given an implicit feedback dataset $\mathcal{R}$ (i.e., a set of 'positive' instances such as clicks or purchases), and the corresponding set of reviews $\mathcal{T}$, we focus on two tasks: item recommendation and review generation. We want to recommend a list of items to a user while also generating plausible reviews that the user might write.

3.1 Collaborative Filtering with Reviews

For the task of item recommendation with implicit feedback, our goal is to estimate pairwise preferences of a user u toward an item i via a scoring function $\hat{y}_{u,i}$. Many state-of-the-art techniques define their predictor in terms of matrix factorization (MF), i.e.,

Table 1: Notation

Notation	Description
$\mathcal{R}, \mathcal{T}$	feedback set, review set
$\mathcal{U}$,$\mathcal{I}$	user set, item set
I^+, I^-	set of observed and unobserved entries
$\hat{y}_{ui}$	predicted 'score' for user u and item i
γ_u, γ_i	latent factors of user u and item i ($K \times 1$)
θ_u, θ_i	text factors for user u and item i ($K \times 1$)
f_u, f_i	word2vec embeddings of u and i ($D \times 1$)
E_u	$K \times D$ embedding matrix of user u
E_i	$K \times D$ embedding matrix of item i
Θ	set of neural network parameters
Φ	set of collaborative filtering parameters

$$\hat{y}_{ui} = \gamma_u^T \gamma_i, \tag{1}$$

where γ_u and γ_i represent K-dimensional user and item latent factors, whose inner product models the preference of u toward i.

There have been many extensions of standard MF methods by incorporating 'side information' such as categorical attributes, image features (He and McAuley, 2016), and text (Hu, 2017; Zheng et al., 2017). To fully take advantage of review text, we start by incorporating text embeddings from reviews into our predictor. Following this we use a single layer MLP to model the interactions between latent factors and text embeddings. The extended predicator is defined as

$$\hat{y}_{ui} = \sigma(\mathbf{W} \begin{bmatrix} \gamma_u^T \gamma_i \\ \theta_u^T \mathrm{E_i} f_i \\ \theta_i^T \mathrm{E_u} f_u \end{bmatrix} + \mathbf{b}), \tag{2}$$

where θ_u and θ_i are K-dimensional (latent) text factors that interact with the text embeddings of u and i. f_u and f_i are D-dimensional text features extracted from reviews written by user u or about item i. Only reviews in the training set are used (since reviews are not available at test time). Specifically, we first train a word2vec model on all reviews in the training set; then we aggregate reviews written by u or about i, and take an average over the representations. $\mathrm{E_u}$ and $\mathrm{E_i}$ are transform matrices that project D-dimensional text features into K-dimensional 'preference space.'

Given the preference predictor, we can learn the model by minimizing the point-wise classification loss (He et al., 2017):

$$\mathcal{L} = -\sum_{(u,i)\in\mathcal{I}^- \cup \mathcal{I}^+} y_{ui} \log \hat{y}_{ui} + (1-y_{ui}) \log(1-\hat{y}_{ui}), \tag{3}$$

where I^+ and I^- represent the set of observed and unobserved entries in $\mathcal{R}$. With (eq. 3), the item recommendation task becomes a classification problem of deciding whether i belongs to the positive or negative feedback set (I^+ or I^-).

3.2 Generative Concatenative Network

Recurrent neural networks with units such as long short term memory (LSTM) and gated recurrent units (GRU) have been widely used as generative models for tasks such as natural language generation, image captioning, dialogue generation, etc. (Józefowicz et al., 2016; Vinyals et al., 2015; Ghosh et al., 2017; Sutskever et al., 2014; Karpathy and Fei-Fei, 2017). In our model, we use a two-layer LSTM network to generate review text at the character level. We adopt the LSTM architecture from Zaremba et al. (2014). Updates of the LSTM layer are defined as:

$$\begin{pmatrix}\mathbf{g}\\ \mathbf{i}\\ \mathbf{f}\\ \mathbf{o}\end{pmatrix} = \begin{pmatrix}\tanh\\ \mathrm{sigmoid}\\ \mathrm{sigmoid}\\ \mathrm{sigmoid}\end{pmatrix} W_l \begin{pmatrix}\mathbf{h}_{l-1}^t\\ \mathbf{h}_l^{t-1}\end{pmatrix}$$
$$\mathbf{s}_l^t = \mathbf{g}_l^t \odot \mathbf{i}_l^t + \mathbf{s}_l^{t-1} \odot \mathbf{f}_l^t$$
$$\mathbf{h}_l^t = \tanh(\mathbf{s}_l^t) \odot \mathbf{o}_l^t, \tag{4}$$

where $\mathbf{s}$ is the internal state of the cell; $\mathbf{g}$ represents the input node which has an activation function; $\mathbf{i}$, $\mathbf{o}$ and $\mathbf{f}$ are sigmoid gating units ('input,' 'output,' and 'forget,' respectively); and $\mathbf{h}$ is the hidden state of the cell. $\tanh$ and $\mathrm{sigmoid}$ are applied element-wise, as is the product $\odot$. W_l is the weight matrix for layer l.

Next we consider the text generation task using 'vanilla' LSTMs. For a character-LSTM model, each input x_t is a character of the original text. Given a text sequence $x^1 \dots x^T$, the LSTM-based network takes an input x^t for each step t and updates its hidden states $\mathbf{h}^t$ based on both the current input x^t and the previous step's hidden state $\mathbf{h}^{t-1}$. The network predicts the input x^{t+1} at the next step, given all inputs $x^{\leq t}$ before $t+1$. The output layer is connected to a softmax layer:

$$p(x^t|x^{<t}, \Theta) = \mathrm{softmax}(W_s \mathbf{h}_L^t), \tag{5}$$

where W_s is the weight matrix of the softmax layer, $\mathbf{h}_L^t$ is the hidden state at step t of the last hidden layer L, and Θ is the complete set of neural network parameters. Given the output probability distribution over all characters, we can predict the output by taking the character that maximizes the probability.

So far, the above method samples text from a 'background' distribution, but lacks the ability to generate personalized and item-specific text. We aim to address this by providing the network with auxiliary inputs x_{aux}. Here we hope that the same user and item latent factors γ_u and γ_i can also be effective as input to the generative model; this expectation is based on the notion that these factors describe the 'aspects' of user preferences and item properties that contribute to the user's overall opinion, and therefore the language in their review. Furthermore the use of low-dimensional latent factors means that the method can straightforwardly scale to large populations of users and items, which is not possible for methods based on one-hot encodings. Following Lipton et al. (2015) we adopt a simple replication strategy to concatenate the latent factors with the character input, So that the input to the model at each step becomes $x'^t = [x_{char}^t; \gamma_u; \gamma_i]$.

By concatenating the latent factors together with the character input, the auxiliary signal is preserved through hundreds of steps, allowing long and coherent sequences to be generated. If we were instead to treat latent factors as inputs to the hidden cell, the signal would quickly vanish or explode. In practice we found that a dimensionality of around $K = 8$ to 50 leads to acceptable performance when modeling users and items. This is a relatively small number compared with the hidden unit size (typically 256-1024), requiring only a modest computational overhead.

By sharing the same user and item latent factors, the tasks of item recommendation and review generation are learned jointly. The complete network structure is shown in Figure 1. We train the model in an end-to-end manner by optimizing the joint loss function:

$$\begin{aligned}\mathcal{L} = &-\sum_{(u,i)\in\mathcal{I}^-\cup\mathcal{I}^+} y_{ui}\log\hat{y}_{ui} + (1-y_{ui})\log(1-\hat{y}_{ui})\\ &- w\sum_{(u,i)\in\mathcal{T}}\sum_{t=1}^{T}\log p(x^t|x^{<t},\Theta,\gamma_u,\gamma_i)\\ &+ \lambda(\|\Theta\|_2^2 + \|\Phi\|_2^2),\end{aligned} \tag{6}$$

where w is a hyperparameter that trades off between the two tasks, $\Phi = \{\gamma_u, \gamma_i, \theta_u, \theta_i, \mathrm{E}_u, \mathrm{E}_i\}$ is the set of collaborative filtering parameters, and λ is a regularization hyperparameter.

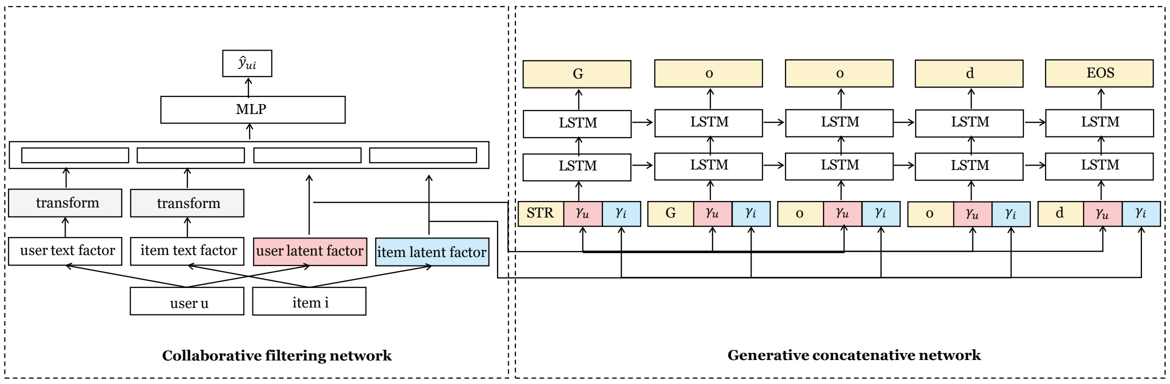


Figure 1: CF-GCN network structure

Table 2: Statistics after pre-processing.

Dataset	#users	#items	#reviews
BeerAdvocate	7,354	8,832	1,270,650
Electronics	20,247	11,589	306,899
Yelp	15,806	12,824	740,984

4 Experiments

We conduct experiments on multiple real-world datasets to investigate the following questions:

- **RQ1**: What can the model learn from review text? Does our joint training framework improve item recommendation performance?
- **RQ2**: What is the fidelity of the generative model? Does it successfully capture user/item attributes, sentiment, and writing style?
- **RQ3**: Can the model be used for personalized review ranking? How does performance improve as more reviews are available during training?

4.1 Datasets

We focus on three real-world datasets: BeerAdvocate,[1] Amazon Electronics,[2] and Yelp.[3] We discard users and items with few actions by extracting the k-core, with $k = 20$ for BeerAdvocate and Yelp, and $k = 10$ for Amazon Electronics (which is substantially sparser). The statistics of the datasets after pre-processing are shown in Table 2.

[1] https://snap.stanford.edu/data/
[2] http://jmcauley.ucsd.edu/data/amazon/
[3] https://www.yelp.com/dataset_challenge

4.2 Experimental Setting

We consider two experimental settings for sampling implicit feedback instances and reviews. In one ('subset') we consider a subset of implicit feedback instances in $\mathcal{R}$ such that each is associated with exactly one review in $\mathcal{T}$; in the second ('wholeset') we consider *all* implicit feedback instances in $\mathcal{R}$, but only a subset of the reviews. In other words, the two settings have the same subset of reviews, but differ in the amount of implicit feedback used. Our reasons for considering the latter setting are twofold: First, training LSTM models with review data is time-consuming, but it is relatively inexpensive to add additional implicit feedback instances during training, meaning that we can achieve a boost in performance with only a modest increase in running time. And second, in real recommendation scenarios, most users do not write reviews, while all provide implicit feedback through their actions; thus this setup allows us to train on both types of feedback simultaneously, and even to model likely reviews for users who have written very few (or none!).

We randomly sample 10%/50%/20% of instances from BeerAdvocate/Electronics/Yelp, respectively. We use a different sample ratio so that the subset maintains around 100,000 reviews for each review model. Then we use a skip-gram model to train word2vec on the training set for each dataset so as to reduce noise and learn better representations for those words that are dataset-specific. We choose an embedding size of 64 and keep the most frequent 20,000 words in each training corpus.

We implemented the proposed method using Keras. For all datasets, we randomly withhold two

Table 3: Comparisons of Hit Rate, NDCG and AUC on three datasets. CF-GCN improves all metrics over BPR and GMF for all settings (higher is better for all metrics).

Dataset	Setting	Hit Rate			NDCG			AUC		
		BPR	GMF	CF-GCN	BPR	GMF	CF-GCN	BPR	GMF	CF-GCN
BeerAdvocate	subset	0.583	0.584	**0.613**	0.351	0.334	**0.371**	0.826	0.847	**0.861**
	wholeset	0.752	0.763	**0.773**	0.476	0.487	**0.501**	0.925	0.925	**0.928**
Electronics	subset	0.375	0.428	**0.459**	0.224	0.254	**0.275**	0.690	0.746	**0.779**
	wholeset	0.494	0.521	**0.529**	0.295	0.317	**0.324**	0.665	0.824	**0.826**
Yelp	subset	0.641	0.660	**0.679**	0.378	0.392	**0.412**	0.899	0.895	**0.902**
	wholeset	0.811	0.830	**0.847**	0.514	0.530	**0.553**	0.946	0.946	**0.952**

interactions per user as our validation/test set, and use all other interactions for training, (i.e., leave-one-out evaluation (Rendle et al., 2009)). All results are reported on the test set, for the hyper-parameters resulting in the best performance on the validation set. The dimensionality of the latent factors is set to $K = 8$. Parameters of the user and item latent factors and text embeddings are initialized from a Gaussian distribution with $\mu = 0$ and $\sigma = 0.01$. For the review model, we stacked two LSTM layers with 256 hidden units per layer. We first train a conventional character-LSTM model then load the pre-trained parameters as initialization to speed up training. During training, we concatenate all reviews in the training set and add start and end tokens (e.g. STR, EOS) as delimiters. Then we split the concatenated string into sequences of length 200.

The model is trained in min-batches with batch-size 256. We adopt the Adam Optimizer (Kingma and Ba, 2014) to update weights. We consider learning rates in $\{0.01, 0.001, 0.0001\}$ and $\lambda \in \{0, 0.001, 0.0001, 0.00001, 0.000001\}$, selecting the optimal values on the validation set using grid search. We also tune the weight w between 0 to 1 to trade off item recommendation vs. review generation in (eq. 6).

4.3 Item Recommendation (RQ1)

We first demonstrate that our model can learn user preferences through both implicit feedback and review text. We focus on the item recommendation task and compare the proposed model with two state-of-the-art methods: Bayesian Personalized Ranking (BPR) (Rendle et al., 2009) and Generalized Matrix Factorization (GMF) (He et al., 2017). Both BPR and GMF use the predicator as eq. 1. We consider three metrics: Hit Rate, NDCG, and AUC. At test time, given a user and item pair from the held-out data, we return a truncated ranked list of items with size 10. Then the Hit Rate at the top 10 (HR@10) represents the ratio of ground truth items existing in the ranked list, while the NDCG measures the position of the hits (He et al., 2017, 2016). The AUC (*Area Under the ROC curve*) is evaluated as:

$$\text{AUC} = \frac{1}{|\mathcal{U}|}\sum_{u}\frac{1}{|\mathcal{S}(u)|}\sum_{(i,j)\in\mathcal{S}(u)}\delta(\hat{y}_{ui} > \hat{y}_{uj}),$$

where δ is an indicator (1 iff its argument is true), and the set of pairs of u is defined as:

$$\mathcal{S}(u) = \{(i,j)|i \in \mathcal{I}^{+}_{test}(u) \\ \wedge j \notin \mathcal{I}^{-}_{train}(u) \cup \mathcal{I}^{-}_{valid}(u) \cup \mathcal{I}^{-}_{test}(u)\}.$$

As shown in Table 3, CF-GCN improves over BPR and GMF on the three metrics. Note that we only use a modest number of latent factors (8) and word2vec embedding dimensions (64). Moreover, we can see that CF-GCN achieves greater improvements in sparse settings, confirming the ability of review text to overcome the sparsity of implicit feedback datasets.

4.3.1 User Cold-Start

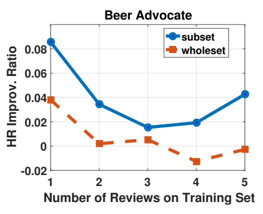
Real recommender systems often suffer from cold-start issues due to data sparsity. It is a critical task for the system to capture user preferences toward items given limited data. To explore the performance of CF-GCN, we further analyze the improvement on cold (or 'cool') users. Figure 2 shows the improvement (in Hit Rate and NDCG) for users with fewer than 5 reviews on BeerAdvocate. The largest improvement ratio happens for users with only a single review during training.

4.4 Review Generation Analysis (RQ2)

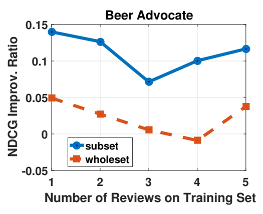
4.4.1 Perplexity

Perplexity is commonly used to measure the quality of generated text. It is defined as

$$\text{ppx} = e^{-\frac{1}{N}\sum_{(u,i)}\frac{1}{T}\sum_{c=1}^{T}\log p(c_t|c_{<t},\Theta,\gamma_u,\gamma_i)},$$



(a) Hit Rate



(b) NDCG

Figure 2: Improvement Ratio on Cold-Start Users

Table 4: Comparison of Perplexity on test set (lower is better).

Dataset	Character LSTM	CF-GCN (subset)	CF-GCN (wholeset)
BeerAdvocate	2.370	2.318	2.329
Electronics	3.033	2.998	2.959
Yelp	2.916	2.817	2.809

where (u, i) are pairs from the test set of reviews $\mathcal{T}_{test}$, N is the number of reviews on the test set, and T is the number of characters in each review.

We compare the test-set perplexity of CF-GCN to that of a standard character-LSTM. As shown in Table 4, CF-GCN achieves lower perplexity than an unsupervised character-LSTM, suggesting that the model is successfully able to leverage information encoded in the user and item representations.

4.4.2 Generated Examples

Figure 3 gives a representative example of a generated review, for a (user,item) pair that is not present in the training set. In other words, only the user and item IDs are provided to the model, from which it must synthesize a plausible review.

Qualitatively, CF-GCN appears to capture the user's writing style, the item's attributes, and the user's sentiment, and more importantly maintains the overall flow and structure of the review in spite of its length and complexity. For example, the synthetic review captures idiosyncrasies of the real review, including the user's tendency to use abbreviations to denote different sensory aspects ('A:' for Appearance, 'S:' for Smell, etc.) CF-GCN also provides descriptions of the item's color, taste and category ("Pours a very dark brown," "a slight sweetness," "bitter," "ale") which approximately match those of the real review. Moreover, the synthetic review shares the same overall sentiment with the real review (3 stars, 'not bad').

Figure 4 presents synthetic review examples conditioned on users who have not written *any* reviews in the training set. We include the ground-truth rating, which is a score between 1.0 and 5.0. The model appears to successfully generates plausible reviews that express sentiment matching the ground-truth rating.

Note here that we only use two LSTM layers with 256 hidden units. Performance could be further improved by increasing the number of hidden units and stacking more layers, at the cost of additional computational resources. For the current setting, it takes around 80 minutes to run 1 epoch for the BeerAdvocate dataset. Typically CF-GCN needs 5 to 10 epochs to converge.

4.5 Personalized review ranking (RQ3)

Besides synthesizing reviews, a more realistic application of our model is to recommend (or rank) existing reviews, by surfacing the review that is most consistent with a user's preferences or writing style. As with our previous experiments, this can be done even for users who have never written a review before: that is, based on their past behavior we can surface the types of reviews written by users who behave similarly.

Since CF-GCN captures both the sentiment and writing style of users, it can also be used in a discriminative way to identify which existing review of an item a particular user is most likely to have written. Given a target item and a list of users, the personalized review ranking problem can be quantitatively evaluated in terms of our ability to rank the 'true' review (i.e., the one the user actually wrote) higher than others. This target captures whether the model can recommend the most relevant existing review to a potential user.

For a review written by user u on item i, we first calculate the perplexity when the generative model is given the latent factors of u and i. We then estimate the AUC by randomly sampling reviews (of the same item) by other users u' and calculating their perplexity when the model is again given the latent factors of u and i. We can then evaluate the model in terms of its ability to associate a lower perplexity to the review u actually wrote. The model generates reasonable results given only a modest number of reviews, and can identify authors with almost perfect accuracy once sufficiently many reviews are provided during training.

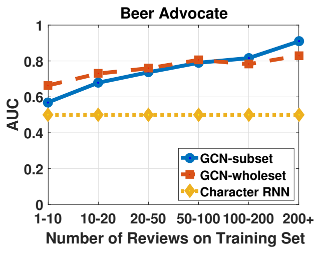
Figure 5 shows AUC results for the personalized review ranking task. As the number of reviews available during training increases, the AUC

Real review	Synthetic review
12 oz. bottle, excited to see a new Victory product around, **A:** Pours a dark brown, much darker than I thought it would be, rich creamy head, with light lace. **S:** Dark cedar/pine nose with some dark bread/pumpernickel. **T:** This ale certainly has a lot of malt, bordering on Barleywine. Molasses, sweet maple with a clear bitter melon/white grapefruit hop flavour. Not a lot of complexity in the hops here for me. Booze is noticable. **M:** Full-bodied, creamy, resinous, nicely done. **D:** A good beer, it isn't exactly what I was expecting. In the end above average, though I found it monotonous at times, hence the 3. A sipper for sure.	**A:** Pours a very dark brown with a nice finger of tan head that produces a small bubble and leaves decent lacing on the glass. **S:** Smells like a nut brown ale. It has a slight sweetness and a bit of a woody note and a little cocoa. The nose is rather malty with some chocolate and coffee. The taste is strong but not overwhelmingly sweet. The sweetness is overpowering, but not overwhelming and is a pretty strong bitter finish. **M:** Medium bodied with a slightly thin feel. **D:** A good tasting beer. Not bad.

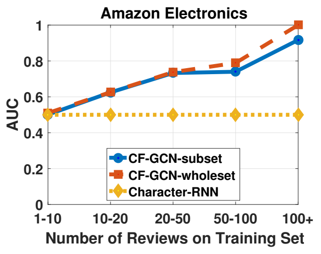
Figure 3: Real and synthetic reviews conditioned on the latent factor of user *Halcyondays* and item *Yakima Glory*. Colors added for emphasis. The model successfully captures the high-level structure of the review, as well as the fine-grained characteristics of their opinion.

Synthetic review: user *BeerShirts*, item *Black Albert*, sentiment *5.0*
A - Pours a brownish-copper color with a thick tan head. S - Strong and smoky aroma, fruity and sweet. The alcohol is not very prominent on the nose. Smells like a stout and enjoyable. It's almost like a hefeweizen and the smell is dominated by a slight burnt sugar character and a slight chocolate smell to it. The taste is sweet and malty. The mouthfeel is very light and watery. Overall a very nice beer that I'd have again.
Synthetic review: user *Morbo*, item *Avalon Spiced Ale*, sentiment *1.5*
I don't like this beer. I was not a big fan of this beer. The aroma was pretty accessible and the taste was a bit sweet and fruity. There was a hint of sweetness that was overwhelmed by the rice and malt also a bit of the hops in the finish. The mouthfeel was medium bodied with a medium carbonation. This was a good beer to drink but I would not be able to get more of this.

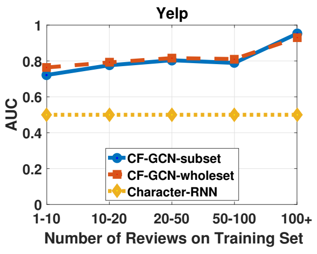
Figure 4: Synthetic reviews for users who have written *no* reviews in the training set. The model successfully predicts user preferences and generates plausible reviews.



(a) BeerAdvocate



(b) Electronics



(c) Yelp

Figure 5: Comparisons of AUC for personalized review ranking for users with different number of reviews on the training set.

also increases, indicating that the method is better able to capture the characteristics of the user and item given more observations.

5 Conclusions

In this paper, we proposed to combine collaborative filtering with generative concatenative networks to jointly perform the tasks of item recommendation and review generation. We formulated the item recommendation task using matrix factorization, in order to capture low-dimensional user preferences and item properties, which we combined with a character-level LSTM model, so that the latent factors are simultaneously responsible for explaining both language and preferences. We adopted a simple input replication strategy to allow the LSTM model to 'remember' the input on which it is conditioned, so that it is able to generate long reviews that capture high-level structure as well as fine-grained sentiment. In addition to using the model generatively, we showed that it can also improve recommendation performance, both in terms of predicting products that a user is likely to interact with, as well as personalized review ranking.

References

Yang Bao, Hui Fang, and Jie Zhang. 2014. Topicmf: Simultaneously exploiting ratings and reviews for recommendation. In *AAAI Conference on Artificial Intelligence*.

Rose Catherine and William W. Cohen. 2017. Transnets: Learning to transform for recommendation. *CoRR*, abs/1704.02298.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Association for Computational Linguistics*.

Sayan Ghosh, Mathieu Chollet, Eugene Laksana, Louis-Philippe Morency, and Stefan Scherer. 2017. Affect-lm: A neural language model for customizable affective text generation. *CoRR*, abs/1704.06851.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850.

Ruining He and Julian McAuley. 2016. Vbpr: Visual bayesian personalized ranking from implicit feedback. In *AAAI Conference on Artificial Intelligence*.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *World Wide Web*.

Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *SIGIR*.

Guang-Neng Hu. 2017. Integrating reviews into personalized ranking for cold start recommendation. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. 2017. Controllable text generation. *CoRR*, abs/1703.00955.

Rafal Józefowicz, Oriol Vinyals, Mike Schuster, Noam Shazeer, and Yonghui Wu. 2016. Exploring the limits of language modeling. *CoRR*, abs/1602.02410.

Andrej Karpathy and Li Fei-Fei. 2017. Deep visual-semantic alignments for generating image descriptions. *IEEE Trans. Pattern Anal. Mach. Intell.*, 39(4):664–676.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*.

Piji Li, Zihao Wang, Zhaochun Ren, Lidong Bing, and Wai Lam. 2017. Neural rating regression with abstractive tips generation for recommendation. In *SIGIR*.

Guang Ling, Michael R. Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *ACM Conference on Recommender Systems*.

Zachary Chase Lipton, Sharad Vikram, and Julian McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *CoRR*, abs/1511.03683.

Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *ACM Conference on Recommender Systems*.

Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Knowledge Discovery and Data Mining*.

Alec Radford, Rafal Józefowicz, and Ilya Sutskever. 2017. Learning to generate reviews and discovering sentiment. *CoRR*, abs/1704.01444.

Steffen Rendle. 2010. Factorization machines. In *International Conference on Data Mining*.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. In *Uncertainty in Artificial Intelligence*.

Ilya Sutskever, James Martens, and Geoffrey Hinton. 2011. Generating text with recurrent neural networks. In *International Conference on Machine Learning*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *International Conference on Neural Information Processing Systems*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *IEEE Conference on Computer Vision and Pattern Recognition*.

Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *Knowledge Discovery and Data Mining*.

Chao-Yuan Wu, Amr Ahmed, and Alexander J. Smola Alex Beutel. 2017. Joint training of ratings and reviews with recurrent recommender networks. In *ICLR Workshops*.

Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. 2014. Recurrent neural network regularization. *CoRR*, abs/1409.2329.

Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *CoRR*, abs/1502.01710.

Lei Zheng, Vahid Noroozi, and Philip S. Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *Web Search and Data Mining*.