Review Sentiment–Guided Scalable Deep Recommender System

Dongmin Hyun[1] Chanyoung Park[1] Min-Chul Yang[2] Ilhyeon Song[2] Jung-Tae Lee[2] Hwanjo Yu[1*]
[1]Dept. of Computer Science and Engineering, POSTECH, Pohang, South Korea
[2]NAVER Corporation, Seongnam, South Korea
{dm.hyun,pcy1302,hwanjoyu}@postech.ac.kr,{minchul.yang,ilhyeon.song,jungtae.lee}@navercorp.com

ABSTRACT

Existing review-aware recommendation methods represent users (or items) through the concatenation of the reviews written by (or for) them, and depend entirely on convolutional neural networks (CNNs) to extract meaningful features for modeling users (or items). However, understanding reviews based only on the raw words of reviews is challenging because of the inherent ambiguity contained in them originated from the users' different tendency in writing. Moreover, it is inefficient in time and memory to model users/items by the concatenation of their associated reviews owing to considerably large inputs to CNNs. In this work, we present a scalable review-aware recommendation method, called SentiRec, that is guided to incorporate the sentiments of reviews when modeling the users and the items. SentiRec is a two-step approach composed of the first step that includes the encoding of each review into a fixed-size review vector that is trained to embody the sentiment of the review, followed by the second step that generates recommendations based on the vector-encoded reviews. Through our experiments, we show that SentiRec not only outperforms the existing review-aware methods, but also drastically reduces the training time and the memory usage. We also conduct a qualitative evaluation on the vector-encoded reviews trained by SentiRec to demonstrate that the overall sentiments are indeed encoded therein.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Recommender System, Deep learning, Sentiment analysis

1 INTRODUCTION

According to the recent technical report from Amazon.com [13], estimated 30% of their page views were from recommendations. As a consequence, a plethora of research has been devoted to building successful recommender systems. Among various recommendation techniques, the most successful approach is collaborative filtering (CF) [5]; it recommends items to a user based on previous ratings of other users whose tastes are similar to the target user. However, this in turn implies that the performance of CF will suffer without a sufficient amount of ratings previously given by users, which is common in reality.

To compensate for the sparsity of the user–item rating data, side information related to users and items, such as user social network [9], user review documents [12, 14–16], and item affinity network [10] has been actively leveraged. In this work, we specifically focus on user review-aware recommendation. User reviews are particularly useful for alleviating the sparsity of user ratings, because the reviews not only embody a user's intention behind the ratings, but also contain conspicuous item properties. That is to say, if reviews are fully exploited, we can build recommender systems even with few ratings provided, which naturally alleviates the sparsity of user–item rating data.

To extract meaningful features from review documents, deep learning-based approaches have been recently proposed [1, 15]. More specifically, convolutional neural network (CNN)-based recommendation methods have gained attention [4, 12, 16] thanks to the capability of CNNs to capture general contextual features from documents. DeepCoNN [16] adopts two CNNs, where one of them models users through reviews written by the users, while the other models items through reviews written for the items. Building upon DeepCoNN, Seo *et al.* propose D-Attn [12] that further adopts the dual local and global attention mechanism on the CNNs, which endow the recommender systems with interpretability regarding the reviews that are used for modeling users and items.

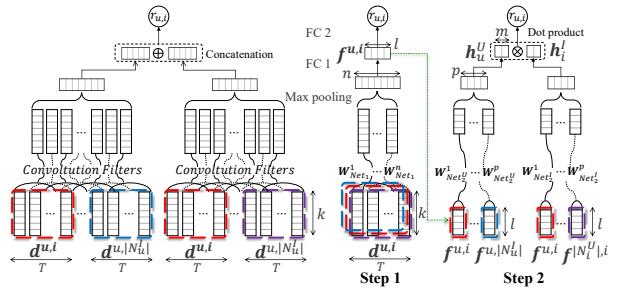
Despite their state-of-the-art performance, they are limited in that users and items are modeled by the reviews consisting of raw words. However, each user has different tendency in writing a review and thus words contain an inherent ambiguity, which makes it hard to precisely understand the user's intent. As a concrete example, let's assume that two different users provided reviews that contain the following identical sentence: "... I like the laptop... ". Whereas a tolerant user would use the word "like" to describe an adequate laptop, a critical user would not use it unless he is completely satisfied with the laptop. However, the previous review-aware methods simply aggregate all the associated reviews and feed them to CNNs expecting the CNNs to automatically extract meaningful features for modeling users and items, which does not suffice for precisely modeling the users and items. This phenomenon compounds when users provided only a few reviews, i.e., cold-start [11], which is common in reality. Moreover, as the existing approaches model each user/item by the concatenation of all the words from every associated review, the size of input for CNNs becomes considerably large, which makes the above approaches practically not feasible in the real-world applications.

In this paper, to overcome the above limitations of the existing methods, we propose a novel sentiment guided review-aware recommendation method, called SentiRec. The core idea is to *leverage*

*Corresponding author

SIGIR '18, July 8–12, 2018, Ann Arbor, MI, USA

ACM ISBN 978-1-4503-5657-2/18/07…$15.00
https://doi.org/10.1145/3209978.3210111



(a) The underlying architecture of baseline methods (b) Architecture of SentiRec

Figure 1: Comparisons of architectures between the baseline methods vs. SentiRec[1].

the overall sentiments of reviews that are represented as ratings that accompany the reviews. In our previous example, if we have a prior knowledge that the tolerant user gave a 3-star rating to the laptop while the critical user gave a 5-star rating, we will be able to more accurately understand the review, which in turn enables us to better model users and items.

Our proposed method consists of two steps. In the first step, instead of representing a review by the concatenation of its constituent raw words as in the previous methods, we encode each review into a fixed-size review vector that is guided to embody the sentiment information of the review. More precisely, we regard a rating that accompanies a review as a summarization of the overall sentiment of a user on an item, and train a CNN that is designed to predict the rating given the review as input, after which a fixed-size vector for the review is obtained by taking the output of the last hidden layer. The second step resembles the training process of DeepCoNN and D-Attn, but is distinguished in that users/items in SentiRec are represented by the concatenation of their associated fixed-size review vectors, rather than raw words. The advantages of SentiRec compared with the previous methods are: 1) we obtain more accurate representations for reviews by incorporating users' overall sentiments on items into reviews, which removes the possible ambiguity contained in the reviews. This in turn results in a better understanding of the reviews, and leads to more accurate representations for users and items resulting in an improved recommendation accuracy. Moreover, 2) we drastically reduce the size of the input, which gives us scalability in terms of the training time and the memory usage. Our experiments show that SentiRec outperforms the state-of-the-art baselines, while being considerably more efficient. Moreover, we perform a qualitative evaluation on the review vectors trained by SentiRec to ascertain that the overall sentiments are indeed encoded in the vectors.

2 BACKGROUND

In this section, we explain how the existing review-aware recommendation methods, i.e., DeepCoNN [16] and D-Attn [12], represent users and items. Note that we assume every rating accompanies a review. Given a review document $\mathbf{d}^{u,i} \in \mathbb{R}^{k \times T}$ on item i written by user u, where k and T denote the latent dimensionality of a word and the average number of words contained in each review, repectively, all the reviews written by user u are concatenated to a single user document matrix $\mathbf{D}_u^U \in \mathbb{R}^{k \times (|N_u^I|T)}$, N_u^I denoting a set of items rated by user u. Likewise, an item document matrix for item i is represented as $\mathbf{D}_i^I \in \mathbb{R}^{k \times (|N_i^U|T)}$, where N_i^U denotes a set of users that rated item i. Then, $\mathbf{D}_u^U$ and $\mathbf{D}_i^I$ are independently fed into two parallel CNNs; one for users and the other for items. After convolutions and max-pooling layers, these CNNs are jointly combined in the last hidden layer to predict the rating of item i given by user u, i.e., $r_{u,i}$. The architecture is shown in Figure 1a.

As mentioned previously, raw words contained in reviews contain an inherent ambiguity owing to users' tendency in writing, and thus entirely depending on the CNNs to extract features that are useful for modeling users and items are prone to error. Moreover, the size of the user/item document matrix $\mathbf{D}_u^U/\mathbf{D}_i^I$ is considerably large making the above methods impractical. Therefore, from the following section, we introduce our two-step approach to overcome the above limitations.

3 METHOD: SentiRec

Step 1: Incorporating Review Sentiments. The goal of this step is to encode each review $\mathbf{d}^{u,i}$ into a fixed-size review vector $\mathbf{f}^{u,i} \in \mathbb{R}^l$, such that the overall sentiment of user u on item i is incorporated. We employ a CNN among various methods [2, 3] to leverage the ratings that accompany the reviews. More precisely, we build a CNN, named $\mathbf{Net}_1$, to predict the rating $r_{u,i} \in \mathbb{R}$ accompanying a review given by user u on item i using the review document $\mathbf{d}^{u,i}$. Formally, $\mathbf{Net}_1$ performs convolution operations on $\mathbf{d}^{u,i}$ using f-th convolution filter $\mathbf{W}^f \in \mathbb{R}^{k \times j}$ with the window size set to j to extract the contextual features c_t^f from the document: $c_t^f = \sigma(\mathbf{W}^f * \mathbf{d}^{u,i}_{(:,(t:t+j-1))} + b^f)$ where $*$ is the convolution operator, $b^f \in \mathbb{R}$ is a bias term for the f-th convolution filter, and σ is a non-linear function such as ReLU. By applying the f-th filter on the entire text $\mathbf{d}^{u,i}$, we obtain a feature map $\mathbf{c}^f_{\mathbf{Net}_1} = [c_1^f, c_2^f, ..., c_t^f, ..., c_{T-j+1}^f]$. Then, max-pooling is applied on the feature map to find the most important feature. i.e., $\hat{c}^f_{\mathbf{Net}_1} = max(\mathbf{c}^f_{\mathbf{Net}_1})$. Finally, with n different convolution filters, followed by a fully connected layer (FC: $\mathbb{R}^n \rightarrow \mathbb{R}^l$), we obtain a vector $\mathbf{f}^{u,i} = \text{FC}([\hat{c}^1_{\mathbf{Net}_1}, \hat{c}^2_{\mathbf{Net}_1}, ..., \hat{c}^n_{\mathbf{Net}_1}]) \in \mathbb{R}^l$, which is passed to a fully connected layer whose output is the predicted rating $r_{u,i}$. Note that $\mathbf{f}^{u,i}$ generated by $\mathbf{Net}_1$ can be regarded as a fixed-size review vector that incorporates the overall sentiment of user u on item i contained in the review document $\mathbf{d}^{u,i}$. The architecture of $\mathbf{Net}_1$ is illustrated in Figure 1b (left).

Step 2: Generating Recommendations. Having trained $\mathbf{Net}_1$ for all the provided reviews, we now proceed to generate the actual recommendations, i.e., rating prediction. Aiming at predicting the rating $r_{u,i}$ that user u will give on item i, we merge all the review vectors of reviews written by user u denoted by $\mathbf{F}_u^U = [\mathbf{f}^{u,i}]_{i \in N_u^I} \in \mathbb{R}^{l \times |N_u^I|}$, and the reviews written for item i denoted by $\mathbf{F}_i^I = [\mathbf{f}^{u,i}]_{u \in N_i^I} \in \mathbb{R}^{l \times |N_i^U|}$. Then we introduce two parallel CNNs; one for modeling users ($\mathbf{Net}_2^U$), and the other for modeling items ($\mathbf{Net}_2^I$). Their network structures are equivalent to that of $\mathbf{Net}_1$, while they only differ in the way the convolution operations are performed. To be precise, as the respective inputs

[1] T can be different for each review as T is the average number of words contained in each review.

Table 1: Data statistics.

Datasets	Office	Grocery	Clothing	Sports
# of users	4,905	14,681	39,387	35,598
# of items	2,418	8,712	23,022	18,351
# of reviews	53,258	151,254	276,677	296,337
Avg. # words / review	168.8	109.9	69.2	99.9
Avg. # reviews / user	8.5	7.9	5.0	6.2
Avg. # reviews / item	17.2	3.3	8.6	12.0
Density	0.45%	0.12%	0.03%	0.05%

$\mathbf{F}_u^U$ and $\mathbf{F}_i^I$ to $\mathbf{Net}_2^U$ and $\mathbf{Net}_2^I$ are concatenated review vectors, the order in which feature vectors are placed is not semantically meaningful; unlike concatenated raw words. Therefore, the window size of the convolution filters of $\mathbf{Net}_2^U$ and $\mathbf{Net}_2^I$ should be fixed to 1 to extract features from each review independently; unlike $\mathbf{Net}_1$ whose window size is set to j for extracting the contextual features from a review document concatenated with raw words. It is important to note that we also update the inputs $\mathbf{F}_u^U$ and $\mathbf{F}_i^I$ during the model training to make the review vectors adapt to the recommendation task. $\mathbf{Net}_2^U$ performs convolution operations on $\mathbf{F}_u^U$ using p convolution filters, which is followed by a max-pooling layer. After a fully connected layer (FC: $\mathbb{R}^p \rightarrow \mathbb{R}^m$), we obtain the final output of $\mathbf{Net}_2^U$ given by $\mathbf{h}_u^U \in \mathbb{R}^m$, and we similarly obtain $\mathbf{h}_i^I \in \mathbb{R}^m$ from $\mathbf{Net}_2^I$. Finally, we calculate a dot product [12] between two vectors $\mathbf{h}_u^U$ and $\mathbf{h}_i^I$, and obtain the predicted rating $r_{u,i} = \langle \mathbf{h}_u^U, \mathbf{h}_i^I \rangle$. We note that only the reviews used to train $\mathbf{Net}_1$ are used to train $\mathbf{Net}_2^U$ and $\mathbf{Net}_2^I$. The architecture of $\mathbf{Net}_2^U$ and $\mathbf{Net}_2^I$ is illustrated in Figure 1b (right).

Summary. In **step 1**, we encode each review into a fixed-size review vector that incorporates the overall sentiment of a user on an item, and leverage this vector-encoded review for modeling users and items in **step 2**. We postulate that the exploitation of the overall sentiment of a user helps us remove the possible ambiguity contained in reviews, and gives us a high-quality representation of each review. Moreover, by representing users and items by the concatenation of their associated fixed-size review vectors rather than raw words, SentiRec obtains scalability; the size of the inputs to $\mathbf{Net}_2^U$ and $\mathbf{Net}_2^I$ is drastically reduced compared with that of the previous review-aware methods. More precisely, the size of the input for user u is reduced from $O(k \times |N_i^U|T)$ to $O(l \times |N_i^U|)$ yielding us at least T times of reduction in terms of the input size[2], which is significant considering that T, i.e., the average number of words in a review, is usually large (Refer to Table 1).

4 EXPERIMENTS

Datasets. We evaluate our proposed method on four real-world datasets extracted from *Amazon.com* by McAuley *et al.* [8]: Office Products, Clothing Shoes & Jewelry, Grocery & Gourmet Food, and Sports & Outdoors. All the reviews in the datasets accompany user–item ratings (1 to 5). We remove users having fewer than five ratings. Table 1 summarizes the detailed statistics of the datasets.

Baselines.

- **MF** [5]: A model-based CF method that projects users and items into low-dimensional vectors solely based on user–item ratings.
- **DeepCoNN** [16]: A CNN-based review-aware recommendation method that models users and items by their associated reviews.
- **D-Attn** [12]: An extension of DeepCoNN that further employs global and local attentions.

Since DeepCoNN and D-Attn have surpassed other review-aware recommendation methods, such as CTR [14], HFT [7], CDL [15], ConvMF [4], we omit them for brevity.

Evaluation Protocol and Metric. We divide each user's ratings into training/validation/test sets in a 80%/10%/10% split. We also evaluate all methods on *cold-start* setting, where we only test on users with fewer than five ratings in the training dataset. All of the hyperparameters are tuned on the validation set by grid search. The best performing parameters for SentiRec are: $j = 4$, $n = 256$, $l = 50$, $p = 5$ and $m = 10$. As for D-Attn, we employ the best parameters that are reported in the paper [12]. As our focus is on recommendations in terms of rating prediction, we employ the mean squared error (MSE), a metric that has been commonly used for evaluating the performance of user rating prediction on the Amazon datasets [7, 10]. Note that we fix the seed for random initialization of all the methods.

Table 2: Test performance in terms of MSE. (*Imprv.* denotes the improvement of SentiRec vs. the best competitor.)

Dataset	Setting	Method				*Imprv.*
		MF	DCNN	D-Attn	SentiRec	
Office	*All*	0.854	0.801	0.784	**0.763**	2.70%
	Cold-start	1.039	0.981	0.956	**0.910**	4.81%
Grocery	*All*	1.026	1.023	0.988	**0.977**	1.14%
	Cold-start	1.093	1.091	1.064	**1.044**	1.91%
Clothing	*All*	1.209	1.175	1.164	**1.120**	3.76%
	Cold-start	1.241	1.213	1.214	**1.164**	4.04%
Sports	*All*	0.957	0.944	0.902	**0.879**	2.46%
	Cold-start	1.014	0.994	0.966	**0.939**	2.72%

4.1 Performance Analysis

Table 2 shows the test performance of all the methods in terms of MSE. We have the following observations: 1) From the comparisons between MF with the rest, we verify the benefit of leveraging reviews as side information for recommendations. 2) We observe that DeepCoNN is outperformed by D-Attn, which extends DeepCoNN by adopting the attention mechanism. This verifies that both local and global attentions help CNNs to better understand the reviews. Note that we show this comparison here, as it is overlooked by D-Attn [12]. 3) SentiRec shows the best performance among all the baselines. This verifies that encoding the overall sentiments of reviews into fixed-size vectors indeed help us remove possible ambiguity contained in the reviews, which eventually facilitates to more accurately model users and items. 4) The performance improvement of SentiRec is consistently larger under the *cold-start* setting. This implies that **step 1** of SentiRec successfully learns the general representations of reviews by sharing a network throughout all the reviews, which enables **step 2** of SentiRec to model users and items even with a few reviews provided.

4.2 Scalability Analysis

Training time. Table 3 shows the the training time comparisons between SentiRec and D-Attn [12], the best performing baseline. As SentiRec is composed of two steps, we report the training time

[2]Note that in our experiments, $k = 100$ is twice as large as $l = 50$.

Table 3: Training time until convergence in seconds. (Numbers in brackets: num. required epochs until convergence.)

Method	Datasets			
	Office	Grocery	Clothing	Sports
(a) D-Attn	9,018 (6)	12,762 (3)	28,494 (3)	51,900 (5)
(b) SentiRec-Step 1	546 (28)	3,303 (23)	3,104 (20)	6,867 (21)
(c) SentiRec-Step 2	60 (15)	75 (5)	75 (3)	150 (5)
Ratio = $(a\ /\ (b+c))$	14.9	3.8	9.0	7.4

Table 4: GPU memory usage (MB)

Method	Datasets			
	Office	Grocery	Clothing	Sports
(a) D-Attn	5,069	5,227	5,211	5,671
(b) SentiRec-Step 1	835	1,353	1,361	1,361
(c) SentiRec-Step 2	659	887	1,081	1,167
Ratio = $(a\ /\ (b+c))$	3.4	2.3	2.1	2.2

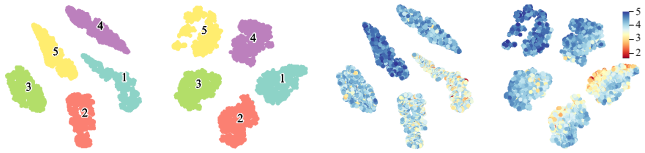
for each of them. We observe that SentiRec trains at most 14.9 times faster than D-Attn. Such improvement is derived from the reduced size of input review documents that are encoded into fixed-size vectors in step 1.

Memory usage. Table 4 shows the GPU memory usage comparisons between SentiRec and D-Attn. We observe at most 3.4 times of improvement of SentiRec over D-Attn. Again, such improvements are due to the reduced input size.

From the performance analysis in Section 4.1 and the scalability analyses in Section 4.2, we verify that SentiRec is a scalable recommendation method that is practical in reality, which even outperforms the state-of-the-art baselines in terms of the recommendation accuracy. It is worth mentioning that the actual training time and the memory usage of SentiRec are shorter and smaller, respectively, because step 1 can be performed off-line in advance.

4.3 Qualitative Analysis

In this section, we aim to qualitatively demonstrate that the overall sentiments of reviews are indeed encoded in the review vectors after training SentiRec. To this end, we perform t-SNE [6] visualizations on the review vectors $\mathbf{f}^{u,i}$ obtained after training step 1 and step 2 of SentiRec. More precisely, each review vector $\mathbf{f}^{u,i}$ is projected to a point, and it is colored once by its associated rating $r_{u,i}$, and another time by the average of the ratings given by all users that rated the item i and the ratings given to all items rated by the user u. i.e., $0.5\times(1/|N_i^U|\sum_{u\in N_i^U} r_{u,i} + 1/|N_u^I|\sum_{i\in N_u^I} r_{u,i})$. The latter one is to determine the general sentiment of user u and item i each associated with $\mathbf{f}^{u,i}$. Figure 2a (left) shows that the review vectors are clearly grouped into their corresponding ratings, verifying that step 1 is correctly trained to reflect the ratings as expected. On the other hand, whereas we observe from Figure 2a (right) that the reviews with similar ratings are still grouped together even after training step 2, *the general sentiments are also revealed in the reviews as shown in Figure 2b (right).* Specifically, when compared with Figure 2b (left), we can clearly see certain trends that the reviews belonging to ratings 1, 2 and 3 are rearranged among themselves according to the general sentiments of the reviews. Such rearrangements are mainly exposed among lower rating gruops (1, 2 and 3), because the general sentiments among each of the higher rating groups (4 and 5) tend to agree among themselves; users and items that give and receive ratings of 5 (Figure 2a (left)) tend to give and receive high



(a) Associated ratings – **Step 1** (left) and **Step 2** (right) (b) Averaged ratings – **Step 1** (left) and **Step 2** (right)

Figure 2: t-SNE visualizations of vector-encoded reviews.

ratings in general (Figure 2b (left)). From the above analysis, we can ascertain that the superior performance of SentiRec is indeed derived from the general sentiments encoded in the review vectors.

5 CONCLUSIONS

We presented SentiRec, a review-aware scalable recommendation method that is guided by the sentiments of reviews. In order to remove the possible ambiguity contained in reviews, we leverage users' overall sentiments on items expressed through the ratings, and encode the reviews into fixed-size vectors. Then, we model users/items by merging their associated reviews by using the vector-encoded reviews, which gives us significant reduction in the training time and the memory usage, followed by two parallel CNNs to generate recommendations. We demonstrate that SentiRec is both effective and efficient compared with the state-of-the-art baselines.

ACKNOWLEDGMENTS

This research was supported by 1) the MSIT, Korea (IITP-2018-2011-1-00783), 2) Basic Science Research Program through the NRF funded by the MSIT (NRF-2017M3C4A7063570), and 3) the NRF grant funded by the MSIT (NRF-2016R1E1A1A01942642).

REFERENCES

[1] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the gru: Multi-task learning for deep text recommendations. In *RecSys*. ACM.
[2] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* (2003).
[3] Yu Chen and Mohammed J Zaki. 2017. Kate: K-competitive autoencoder for text. In *SIGKDD*. ACM.
[4] 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 RecSys*. ACM, 233–240.
[5] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *Computer* 42, 8 (2009).
[6] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* (2008).
[7] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. ACM.
[8] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton Van Den Hengel. 2015. Image-based recommendations on styles and substitutes. In *SIGIR*. ACM.
[9] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. 2016. Trecso: Enhancing top-k recommendation with social information. In *WWW*. 89–90.
[10] Chanyoung Park, Donghyun Kim, Jinoh Oh, and Hwanjo Yu. 2017. Do Also-Viewed Products Help User Rating Prediction?. In *WWW*.
[11] Andrew I Schein, Alexandrin Popescul, Lyle H Ungar, and David M Pennock. 2002. Methods and metrics for cold-start recommendations. In *SIGIR*. ACM.
[12] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Interpretable convolutional neural networks with dual local and global attention for review rating prediction. In *RecSys*. ACM.
[13] Brent Smith and Greg Linden. 2017. Two Decades of Recommender Systems at Amazon. com. *IEEE Internet Computing* (2017).
[14] Chong Wang and David M Blei. 2011. Collaborative topic modeling for recommending scientific articles. In *SIGKDD*. ACM.
[15] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *SIGKDD*. ACM.
[16] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *WSDM*. ACM.