Latent Multi-Criteria Ratings for Recommendations

Pan Li
New York University
44th West 4th Street, NY, US
pli2@stern.nyu.edu

Alexander Tuzhilin
New York University
44th West 4th Street, NY, US
atuzhili@stern.nyu.edu

ABSTRACT

Multi-criteria recommender systems have been increasingly valuable for helping consumers identify the most relevant items based on different dimensions of user experiences. However, previously proposed multi-criteria models did not take into account latent embeddings generated from user reviews, which capture latent semantic relations between users and items. To address these concerns, we utilize variational autoencoders to map user reviews into latent embeddings, which are subsequently compressed into low-dimensional discrete vectors. The resulting compressed vectors constitute latent multi-criteria ratings that we use for the recommendation purposes via standard multi-criteria recommendation methods. We show that the proposed latent multi-criteria rating approach outperforms several baselines significantly and consistently across different datasets and performance evaluation measures.

CCS CONCEPTS

• **Information systems → Information retrieval**.

KEYWORDS

Multi-Criteria Recommendation System, Collaborative Filtering, User Preference, Multi-Criteria Decision Making

ACM Reference Format:
Pan Li and Alexander Tuzhilin. 2019. Latent Multi-Criteria Ratings for Recommendations. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3298689.3347068

1 INTRODUCTION AND RELATED WORK

The field of recommender systems has experienced extensive growth in many research directions over the last decade [2, 18], including the area of multi-criteria recommendations [1, 11, 12, 19]. For example, several prominent websites from Zagat to TripAdvisor collect multi-criteria ratings to measure quality of items shown on their sites that can subsequently be used for recommendation purposes.

Previous researchers try to improve accuracy of multi-criteria recommender systems in various ways, including Support Vector Regression [11] or the halo effect [19]. However, most of the methods do not take the information contained in user reviews into account, which could potentially alleviate the sparsity problem and improve quality of recommendations, as pointed out in [4]. Some researchers propose to utilize aspect information [3, 6, 17, 22] from user reviews for multi-criteria recommendations. Although useful, these papers do not consider the latent semantic information contained in user reviews, which is beneficial for understanding users' true experiences and expectations [23]. Besides, user reviews contain high-dimensional user feedback information beyond aspects, while the multi-criteria ratings collected by the online platforms are limited by low-dimensional pre-defined criteria, which may not fully represent multiplicity of user experiences. Moreover, some of the multi-criteria ratings might be missing, thus limiting performance of explicit multi-criteria rating methods [1].

To address these concerns, it's natural to map the user reviews into latent embeddings and incorporate these embeddings into the recommendation process. However, typical review embeddings are noisy and high-dimensional, which significantly increases the difficulty of computation and optimization. To resolve this issue, prior literature proposed to compress text embeddings with reasonable semantic segmentation into low-dimensional discrete vectors [5, 20], showing that the compressed vectors manage to achieve better performance in sentiment analysis and machine translation tasks. In this paper, we extend this idea from text analysis to multi-criteria recommender systems and propose to use these "compressed vectors" as latent multi-criteria ratings for recommendation purposes.

Specifically, we propose to extract latent multi-criteria ratings from the user reviews using the variational autoencoder [14]. Furthermore, we map the reviews into latent embeddings to represent high-dimensional user experiences, and then perform embedding compression (e.g., using Gumbel-Softmax Reparameterization [10, 16]) to compress them into low-dimension discrete vectors, which constitute the latent multi-criteria ratings for recommendations. We empirically validate the proposed method on three datasets and demonstrate that our approach outperforms several important baselines consistently and significantly in terms of various recommendation accuracy measures.

Note that, the proposed latent multi-criteria ratings method has the following advantages over the method using multi-criteria ratings explicitly provided by the users:

- It is not limited by the pre-defined criteria to model user experiences.
- It does not require to collect multi-criteria feedback from the user.
- It does not need to deal with the missing values problem for multi-criteria ratings.
- It captures latent interactions between users and items, thus providing several benefits reported in [23].

RecSys '19, September 16–20, 2019, Copenhagen, Denmark

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347068

In this paper, we make the following contributions. We propose a novel method that automatically generates latent multi-criteria ratings from user reviews by combining autoencoding and embedding compression techniques for multi-criteria recommendations. We also empirically demonstrate that our approach outperforms the selected baseline models consistently and significantly on three datasets across various experimental settings.

2 METHOD

In this section, we introduce the proposed model for latent multi-criteria recommendations by combining autoencoding and embedding compression techniques, as presented in Figure 1. In Stage 1, we use the variational autoencoder to project the user reviews onto a latent continuous space, and then utilize embedding compression techniques to compress embeddings obtained in the previous stage into discrete latent ratings during Stage 2. Finally in Stage 3, we apply various multi-criteria recommendation algorithms on the latent ratings to produce recommended items.

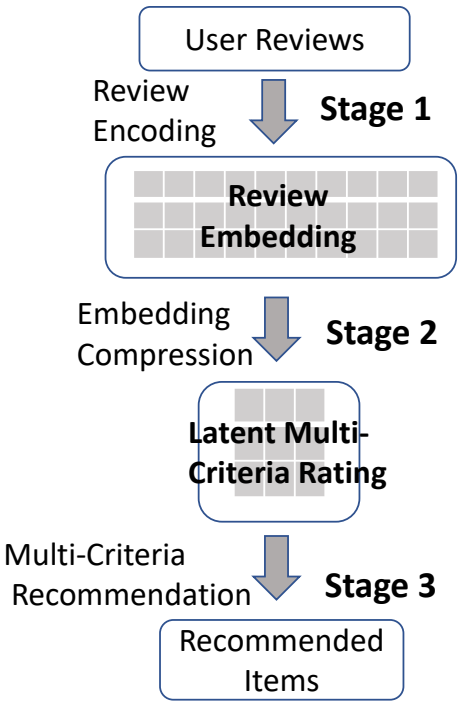

Figure 1: Illustration of Our Proposed Model

2.1 Stage 1: Review Encoding

To project the discontinuous user reviews into latent continuous embeddings, we follow the idea of autoencoding [21] and implement the bidirectional GRU [7] neural networks as the encoder and the decoder respectively. Compared with classical models like RNN or LSTM, GRU is computationally more efficient and better captures semantic meanings [8].

During the training process, every word w in the review $s = \{w^1, w^2, \cdots, w^{N_s}\}$ is mapped to their corresponding word indexes in the pre-defined vocabulary V. Both the input and the output of the constructed autoencoder are the sequences of indexes. To illustrate the GRU learning procedure, we denote $[W^z, W^r, U^z, U^r]$ as the weight matrices of current information and the past information for the update gate and the reset gate respectively. x_t is the index vector input at the timestep t, h_t stands for the output vector, z_t denotes the update gate status and r_t represents the status of reset gate. The hidden state at timestep t could be obtained following these equations:

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z) \quad (1)$$

$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r) \quad (2)$$

$$h_t = (1 - z_t) \circ h_{t-1} + z_t \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h) \quad (3)$$

By iteratively calculating hidden step throughout every time step, we obtain final hidden state at the end of the sentence, which constitutes the review embeddings $R = h_{N_s}$ that captures the latent semantic information of user reviews.

2.2 Stage 2: Embedding Compression

Note that, review embeddings obtained in Stage 1 are noisy and of high-dimensions, which significantly affect the performance of recommendation system. This motivates us to utilize embedding compression methods for efficient learning. We denote that the embedding matrix R consists of $|S|$ embedding vectors with H dimensions, where $|S|$ refers to the total number of reviews in the dataset. Given the arbitrary number K and M, our goal is to find out discrete codes of dimension K ($K << H$) that take integer value ranging from 1 to M in the latent dimension.

Inspired by the Gumbel-Softmax reparameterization method [10, 20], we conduct the embedding compression process using the compositional code learning method that parameterizes the discrete codes onto continuous distributions. The Gumbel-Softmax technique [9, 16] provides a simple and efficient way to get samples z from a categorical distribution with class probabilities π:

$$z = \text{one-hot}(argmax_i[g_i + log\pi_i]) \quad (4)$$

where g_i are drawn from the Gumbel distribution and $one - hot$ stands for the one-hot function that transforms the data to a binary one-hot encoding. We use the softmax function as the continuous differentiable approximation to the argmax function, so that the generated sample vectors would be:

$$y_i = \frac{exp((log(\pi_i) + g_i)/\gamma)}{\sum_{K=1}^{K} exp((log(\pi_K) + g_K)/\gamma)} \quad (5)$$

for $i = 1, 2, \cdots, K$ where γ represents the temperature of the softmax function. Therefore as discussed in [20], if we reverse the entire sampling process described above, we can learn the discrete codes from our continuous embeddings.

Specifically, given the number of latent dimensions K, we first apply the matrix factorization [15] to get the top-K factor for that embedding matrix: $R = \sum_{i=1}^{K} D^i A_i$ where $A_i \in R^{K*H}$ is the basis matrix for the $i - th$ component. Therefore, the learning of discrete codes would be equivalent to the learning of a set one-hot vectors d_w^i so that $\hat{R} = \sum_{i=1}^{K} A_i^T d_w^i$ Furthermore, we assume that the discrete vectors d_w^i are sampled from a prior distribution via

Gumbel-Softmax Reparameterization method [20] by minimizing the reconstruction loss compared with the origin embedding matrix,

$$min_{w_i} \frac{1}{|S|} \sum_{s \in S} ||R_s - \hat{R_s}|| \quad (6)$$

where R'_s stands for the embedding matrix reconstructed from the compressed codes we get. We optimize the prior parameters π_i and g_i. Finally, the compressed vectors for each review embeddings D_w could be obtained by applying $argmax$ to the one-hot vectors d^i_w.

2.3 Stage 3: Multi-Criteria Recommendation

When we compressed continuous review embeddings R into discrete latent embeddings D_w as described in Stage 2, we select the dimension of discrete codes K corresponding to the dimension of collected multi-criteria ratings, and also select the range of each latent dimension M matching with the range of organic multi-criteria ratings. In that sense, we could treat these compressed vectors D_w as the **latent** multi-criteria ratings and then utilize these multi-criteria ratings for recommendation purposes, shown as Stage 3 in Figure 1. Note that, unlike traditional cases, the latent multi-criteria ratings are not specified by the users but obtained from the reviews as described in the previous stages. We conduct the multi-criteria recommendations using the state-of-the-art methods introduced in [1].

To summarize, we propose to uniquely combine the method of variational autoencoders with the embedding compression techniques to learn the latent multi-criteria ratings from user reviews for multi-criteria recommendation purposes. In the next section, we experimentally demonstrate the superiority and effectiveness of the proposed approach.

	Yelp Round 8	Yelp Round 11	TripAdvisor
# Reviews	40,314	11,285	25,928
# Items	1,134	7,788	2,933
# Users	600	587	4,252
Sparsity	5.89%	0.25%	0.21%

Table 1: Descriptive Statistics of Three Datasets

3 EXPERIMENTS AND RESULTS

3.1 Experimental Settings

We implement the model on the following datasets: Yelp Challenge Dataset [1] Round 8 and Round 11 of restaurant recommendations and on TripAdvisor Dataset [2] of hotel recommendations that contain user reviews, as well as multi-criteria feedback. The difference between two Yelp datasets is the time when the data is collected: Round 8 dataset is collected in 2016, while Round 11 dataset is collected in 2018. To avoid the sparsity and the cold start problems, we only consider users that rate at least five items, restaurants that have been rated by at least ten users and hotels that have been rated by at least five users. The basic statistics of the filtered datasets are shown in Table 1.

[1] https://www.yelp.com/dataset
[2] http://www.cs.cmu.edu/~jiweil/html/hotel-review.html

To demonstrate the effectiveness of the proposed latent multi-criteria rating (LatentMC) model, we select several multi-criteria rating models for comparison, including:

- **MC**: the multi-criteria ratings explicitly specified by the users in the three datasets.
- **Overall**: the overall ratings in the three datasets.
- **Embedding**: the uncompressed review embeddings obtained in Stage 1 (see Figure 1) as latent multi-criteria ratings in the three datasets.
- **Aspect**: the extracted aspect ratings [3] as multi-criteria ratings in the three datasets.
- **PCA**: the top-k factors extracted using Principal Component Analysis [13] as latent multi-criteria ratings in the three datasets.

Also, to evaluate recommendation performance, we implement the state-of-the-art multi-criteria recommendation models introduced in [1], including KNN, SlopeOne, CoCluster, Support Vector Regression (SVR) and Aggregate Function. The first three models are adjusted to their multi-criteria recommendation versions by calculating similarities using multi-criteria ratings instead of only overall ratings.

3.2 Experimental Results

We conduct the 5-fold cross-validation recommendation experiments and report the performance results on the test set. As shown in Table 2, our proposed LatentMC model outperforms the baselines consistently and significantly across different datasets, performance measures and algorithms. For example using Aggregate algorithm, the Pre@1, Pre@5, Rec@1, Rec@5 measures for the three datasets improve by 1.73%, 1.56%, 6.05%, 1.29%, 1.91%, 2.32%, 4.03%, 2.00%, 2.52%, 4.86%, 5.47% and 1.99% respectively compared to the second-best baseline models. In general, we obtain over 2% accuracy improvements in most of the cases.

Furthermore, we make the following observations. First, compared with the recommendation results directly using uncompressed review embeddings, the latent compressed multi-criteria rating method achieves better performance. The improvement is achieved through the compression process that cleans up the noisy high-dimensional embedding vectors and extracts the essence of the reviews. Second, we observe that review-based multi-criteria recommendation model performs better that the non-review recommendation model, even without using the explicit multi-criteria rating information. This indicates that user reviews contain richer and high-dimensional information, compared to low-dimensional multi-criteria ratings. Thus, it is crucial to properly model user reviews in the recommendation process. Finally, the latent rating method outperforms the explicit rating models because they capture the latent semantic information within user reviews, which supports the general advantages of using latent methods in text analysis [23].

3.3 Conclusions

To conclude, the latent multi-criteria ratings generated by the proposed model achieve significantly better performance for multi-criteria recommendations in comparison to the alternative methods. In addition, it has the following natural advantages over classical

Algorithm	Rating	Yelp Round 11				Yelp Round 8				TripAdvisor			
		Pre@1	Pre@5	Rec@1	Rec@5	Pre@1	Pre@5	Rec@1	Rec@5	Pre@1	Pre@5	Rec@1	Rec@5
KNN	**LatentMC**	**0.8150***	**0.6822**	**0.2401***	**0.8387***	**0.7275***	**0.6864**	**0.3084***	**0.6907***	**0.7568***	**0.6792**	**0.3472***	**0.7655***
	MC	0.7743	0.6775	0.2199	0.8270	0.7092	0.6741	0.2880	0.6772	0.7298	0.6529	0.3208	0.7536
	Overall	0.7920	0.6750	0.2327	0.8272	0.7078	0.6543	0.2902	0.6659	0.7304	0.6609	0.3232	0.7544
	Embedding	0.7920	0.6720	0.2309	0.8213	0.6978	0.6582	0.2856	0.6709	0.7312	0.6718	0.3220	0.7088
	PCA	0.7918	0.6704	0.2268	0.8000	0.7048	0.6602	0.2875	0.6660	0.7314	0.6788	0.3199	0.7332
	Aspect	0.7977	0.6768	0.2315	0.8305	0.7056	0.6624	0.2844	0.6718	0.7285	0.6042	0.3244	0.7620
SlopeOne	**LatentMC**	**0.8400***	**0.6953***	**0.2382***	**0.8395***	**0.7268***	**0.6947***	**0.3028***	**0.6890**	**0.7563***	**0.6785***	**0.3458***	**0.7675**
	MC	0.7760	0.6777	0.2242	0.8254	0.7078	0.6686	0.2903	0.6742	0.7308	0.6666	0.3276	0.7076
	Overall	0.7920	0.6714	0.2258	0.8210	0.7066	0.6636	0.2912	0.6744	0.7296	0.6624	0.3250	0.7067
	Embedding	0.7663	0.6819	0.2004	0.8253	0.7092	0.6820	0.2880	0.6836	0.7270	0.6066	0.3228	0.5906
	PCA	0.7992	0.6705	0.2205	0.8020	0.7078	0.6692	0.2793	0.6620	0.7195	0.6264	0.3220	0.6880
	Aspect	0.7600	0.6704	0.2088	0.8204	0.7003	0.6838	0.2858	0.6836	0.7285	0.6042	0.3244	0.7620
CoCluster	**LatentMC**	**0.8400***	**0.6880***	**0.2414***	**0.8379***	**0.7292***	**0.6896***	**0.3030***	**0.6930***	**0.7550***	**0.6566**	**0.3468***	**0.7815***
	MC	0.8160	0.6688	0.2295	0.8251	0.7068	0.6642	0.2928	0.6730	0.7332	0.6510	0.3280	0.7628
	Overall	0.7520	0.6736	0.2173	0.8221	0.7076	0.6795	0.2920	0.6830	0.7318	0.6436	0.3288	0.7355
	Embedding	0.8080	0.6784	0.2334	0.8290	0.7024	0.6695	0.2910	0.6795	0.7296	0.6548	0.3302	0.6476
	PCA	0.8114	0.6759	0.2119	0.8000	0.7078	0.6692	0.2793	0.6620	0.7190	0.6464	0.3302	0.6476
	Aspect	0.7823	0.6624	0.2284	0.8214	0.6998	0.6686	0.2902	0.6788	0.7325	0.6166	0.3268	0.7229
SVR	**LatentMC**	**0.8792***	**0.7077***	**0.2984***	**0.8808***	**0.7868***	**0.7890***	**0.3348***	**0.9432**	**0.8382***	**0.9774***	**0.3692***	**0.5392***
	MC	0.8568	0.6894	0.2802	0.8656	0.7692	0.7778	0.3130	0.9333	0.8072	0.9502	0.3380	0.5210
	Overall	0.8608	0.6880	0.2810	0.8670	0.7538	0.7767	0.3098	0.9255	0.8002	0.9530	0.3298	0.5210
	Embedding	0.8590	0.6866	0.2798	0.8656	0.7550	0.7680	0.3104	0.9020	0.7988	0.9530	0.3336	0.5310
	PCA	0.8392	0.6820	0.2798	0.8650	0.7566	0.7660	0.3100	0.8998	0.8002	0.9459	0.3330	0.5310
	Aspect	0.8608	0.6902	0.2815	0.8700	0.7593	0.7756	0.3098	0.9333	0.8036	0.9414	0.3358	0.5175
Aggregate	**LatentMC**	**0.8230***	**0.6928***	**0.2478***	**0.8400***	**0.7532***	**0.7578***	**0.3128***	**0.6988***	**0.6978***	**0.4073***	**0.3532***	**0.7001***
	MC	0.8088	0.6808	0.2302	0.8292	0.7388	0.7402	0.2978	0.6782	0.6767	0.3819	0.3318	0.6862
	Overall	0.8072	0.6792	0.2315	0.8178	0.7388	0.7440	0.2956	0.6664	0.6780	0.3270	0.3306	0.6171
	Embedding	0.8028	0.6820	0.2328	0.8230	0.7402	0.7398	0.3002	0.6792	0.6767	0.3335	0.3298	0.6363
	PCA	0.7978	0.6780	0.2315	0.8230	0.7398	0.7398	0.2988	0.6660	0.6767	0.3330	0.3306	0.6360
	Aspect	0.8078	0.6808	0.2318	0.8208	0.7356	0.7402	0.2972	0.6848	0.6802	0.3875	0.3320	0.6787

Table 2: Experiment results on three datasets. * stands for significance under 95% confidence

multi-criteria recommendation approaches. First, it is not limited by the pre-defined criteria and missing values to model user experiences. Second, it does not require to collect multi-criteria feedback from the user. Third, it captures latent interactions between users and items, which provide several benefits reported in [23].

As the future work, we would like to improve the latent multi-criteria rating generation process even further. Also, we plan to study the semantic meaning and interpretability of the latent multi-criteria ratings.

REFERENCES

[1] Gediminas Adomavicius and YoungOk Kwon. 2015. Multi-criteria recommender systems. In *Recommender systems handbook*. Springer, 847–880.

[2] Gediminas Adomavicius and Alexander Tuzhilin. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge & Data Engineering* 6 (2005), 734–749.

[3] Konstantin Bauman, Bing Liu, and Alexander Tuzhilin. 2017. Aspect based recommendations: Recommending items with the most valuable aspects based on user reviews. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 717–725.

[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] Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2018. Learning K-way D-dimensional Discrete Codes for Compact Embedding Representations. *arXiv preprint arXiv:1806.09464* (2018).

[6] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. *arXiv preprint arXiv:1802.07938* (2018).

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

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

[9] Emil Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series* 33 (1954).

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

[11] Dietmar Jannach, Zeynep Karakaya, and Fatih Gedikli. 2012. Accuracy improvements for multi-criteria recommender systems. In *Proceedings of the 13th ACM conference on electronic commerce*. ACM, 674–689.

[12] Dietmar Jannach, Markus Zanker, and Matthias Fuchs. 2014. Leveraging multi-criteria customer feedback for satisfaction analysis and improved recommendations. *Information Technology & Tourism* 14, 2 (2014), 119–149.

[13] Ian Jolliffe. 2011. *Principal component analysis*. Springer.

[14] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

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

[16] Chris J Maddison, Daniel Tarlow, and Tom Minka. 2014. A* sampling. In *Advances in Neural Information Processing Systems*. 3086–3094.

[17] Cataldo Musto, Marco de Gemmis, Giovanni Semeraro, and Pasquale Lops. 2017. A Multi-criteria Recommender System Exploiting Aspect-based Sentiment Analysis of Users' Reviews. In *Proceedings of the eleventh ACM conference on recommender systems*. ACM, 321–325.

[18] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender systems handbook*. Springer, 1–34.

[19] Nachiketa Sahoo, Ramayya Krishnan, George Duncan, and Jamie Callan. 2012. Research note—the halo effect in multicomponent ratings and its implications for recommender systems: The case of yahoo! movies. *Information Systems Research* 23, 1 (2012), 231–246.

[20] Raphael Shu and Hideki Nakayama. 2017. Compressing Word Embeddings via Deep Compositional Code Learning. *arXiv preprint arXiv:1711.01068* (2017).

[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[22] Hongning Wang, Yue Lu, and Chengxiang Zhai. 2010. Latent aspect rating analysis on review text data: a rating regression approach. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 783–792.

[23] Shuai Zhang, Lina Yao, Aixin Sun, and Yi Tay. 2019. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)* 52, 1 (2019), 5.