A Generative Model for Review-Based Recommendations

Oren Sar Shalom
oren.sarshalom@gmail.com
Intuit AI

Guy Uziel
uziel.guy@gmail.com
IBM Research

Amir Kantor
amirka@il.ibm.com
IBM Research

ABSTRACT

User generated reviews is a highly informative source of information, that has recently gained lots of attention in the recommender systems community. In this work we propose a generative latent variable model that explains both observed ratings and textual reviews. This latent variable model allows to combine any traditional collaborative filtering method, together with any deep learning architecture for text processing. Experimental results on four benchmark datasets demonstrate its superiority comparing to all baseline recommender systems. Furthermore, a running time analysis shows that this approach is in order of magnitude faster that relevant baselines. Moreover, underlying our solution there is a general framework that may be further explored.

CCS CONCEPTS

• **Information systems** → **Collaborative filtering**.

KEYWORDS

Recommender Systems; Collaborative Filtering; User Reviews

ACM Reference Format:
Oren Sar Shalom, Guy Uziel, and Amir Kantor. 2019. A Generative Model for Review-Based Recommendations. In *Thirteenth ACM Conference on Recommender Systems (RecSys '19), September 16–20, 2019, Copenhagen, Denmark.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3298689.3347061

1 INTRODUCTION

Recommender systems play a key role in the web, and have been proven to increase user satisfaction in many domains. Often, these systems model user preference by processing rating and click data generated by the interaction of users with items. Another important signal, that has drawn substantial attention recently, is user generated reviews on items. In this paper, we focus on the integration of textual reviews with explicit numeric ratings. Clearly, this integration can be beneficial since a review usually contains the user's multifaceted impression of the item. While most previous work [4, 31] aim at inferring both user preferences and item traits, our novel approach aims at extracting the *distribution* of the matching between users and items. Hence, it is dubbed as Matching Distribution by Reviews (MDR). Adhering to matching allows to integrate natural language processing (NLP) methods with collaborative filtering (CF) techniques. Thereby, it simplifies the algorithm: when we face a review of a user on an item, their matching inherently spans only a single review; while user preferences or item traits span multiple reviews, which is harder to process.

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

ACM ISBN 978-1-4503-6243-6/19/09…$15.00
https://doi.org/10.1145/3298689.3347061

2 ALGORITHM DETAILS

The input for MDR is a set interactions $D = (u, i, r_{ui}, t_{ui})$. Each element logs an interaction between user u and item i, along with an explicit rating r_{ui} and a textual review t_{ui}.

In the first phase, MDR learns how to find all plausible matching vectors between a user and an item, as expressed in the review. These vectors are modeled by f dimensional vectors over the same space $F = \mathbb{R}^f$. Each dimension in these vectors measures the matching between the user preferences and the item trait regarding the corresponding latent feature. In the second phase of MDR, given these matching vectors, it models both users and items by f dimensional vectors as well. This time the latent dimensions have a different interpretation. The dimensions in the user (item) vectors measure the extent to which the user prefers (the item possesses) these latent features. The first phase can incorporate any text processing network, and the second phase can invoke any CF method. Therefore, our approach is a *general framework*, that can exploit future advances in either NLP or CF. These phases are learned sequentially. First the parameters of the first phase are learned; then these parameters are kept fixed and the parameters of the second phase are learned. We also tried an end-to-end solution, but it led to inferior performance.

2.1 Distribution of Matching Vectors

Given a user generated review on an item, this phase infers the matching between the user and the item across some latent features. Not all textual reviews accommodate modeling in the same manner; for instance, some reviews might be ambiguous or simply do not cover some important factors. Therefore, given a textual review, there could be a varied level of uncertainty in the inferred values of the various latent features. An advantage of our algorithm is the ability to infer the *distribution* of the matching vectors, which is more robust than concrete values. Formally, for each event $(u, i, r_{ui}, t_{ui}) \in D$, it outputs a distribution of matching vectors $m_{ui} \in \mathbb{R}^f$ that indicates the matching between user u and item i over the f latent factors.

We assume a Generative Latent Variable model, in which there is a vector of latent variables z_{ui} that influences the user while generating the observed rating r_{ui} and review t_{ui}. Concretely, z_{ui} and r_{ui} are drawn from probability distributions $P(z_{ui}|t_{ui};\theta)$ and $P(r_{ui}|z_{ui};\theta)$ respectively, parameterized by a vector θ. A consequence of this graphical model, is that r_{ui} and t_{ui} are conditionally independent given z_{ui}. Namely, $P(r_{ui}|z_{ui}, t_{ui};\theta) = P(r_{ui}|z_{ui};\theta)$. To avoid clutter, we omit θ and subscript ui when their existence is clear from the context.

For simplicity, let us assume for a moment that $D = (R, T) = \{r_{ui}, t_{ui}\}$. That is, the input does not contain the identities of the users and the items. Our objective is to maximize the posterior probability, under the generative process:

$$\begin{aligned}&\underset{\theta}{\operatorname{argmax}} P(\theta|D) = \underset{\theta}{\operatorname{argmax}} \frac{1}{P(D)} \cdot P(\theta) \prod_{(r,t)\in D} P(r|t;\theta)P(t;\theta) = \\ &\underset{\theta}{\operatorname{argmax}} P(\theta) \prod_{(r,t)\in D} P(r|t;\theta) = \underset{\theta}{\operatorname{argmax}} \ln P(\theta) + \sum_{(r,t)\in D} \ln P(r|t;\theta)\end{aligned} \tag{1}$$

The second transition holds since our model has no effect on $P(t)$. We now analyze the likelihood term:

$$\begin{aligned}\sum_{(r,t)\in D} \ln P(r|t) &= \sum_{(r,t)\in D} \ln \int_z P(r|z,t)P(z|t)dz = \\ &\sum_{(r,t)\in D} \ln \int_z P(r|z)P(z|t)dz \\ &\geq N \cdot \mathbf{E}_{(r,t)\sim D}[\mathbf{E}_{z\sim P(\cdot|t)}[\ln P(r|z)]].\end{aligned} \tag{2}$$

The first equality follows from the continuous version of the law of total probability. The second holds since $r_{ui} \perp t_{ui} \mid z_{ui}$. The last inequality follows from Jensen's inequality and the concavity of the log function. Thus, we are able to find a lower bound for the objective. We now describe the choices of $P(r|z)$ and $P(z|t)$.

$P(r|z)$: The latent variables z account for a user's decision in the process of generating a review, while a matching vector m simply describes the matching between the user and the item. Therefore, z may be considered as an intermediate step in the process of inferring m. This is done by applying a parametric family of deterministic functions $g(z;\theta) = c$. Our choice of $g(\cdot)$ is a fully connected layer followed by a hyperbolic tangent as a non-linear activation function.

Vector m represents the matching between the user and the item over the latent features. Hence, the sum of its factors $\sum_f c^{[f]}$ aggregates the matching score and may be used to approximate the actual rating, where $c^{[f]}$ stands for the f^{th} entry in vector m. To mimic the effect of the random processes that may happen as part of the rating generation, we assume the approximation error is attributed to some additive white Gaussian noise with variance λ, for some hyper-parameter λ. Combining all, we obtain:

$$P(r|z;\theta) = \mathcal{N}(\sum_f g(z;\theta)^{[f]},\ \lambda)$$

That is, the rating is drawn from a normal distribution with mean $\sum_f g(z;\theta)^{[f]}$ and variance λ.

$P(z|t)$: The proposed model does not make any assumptions on $P(z)$, and only assumes the structure of $P(z|t)$. Our natural choice is to say that $P(z|t) = \mathcal{N}(\mu(t;\theta), \Sigma(t;\theta))$, where $\mu(\cdot)$ and $\Sigma(\cdot)$ are two deterministic functions parameterized by θ. They are implemented via a deep learning architecture, where their parameters are partially shared. Throughout, we will omit θ from these functions for brevity. We further constrain Σ to output diagonal matrices. Therefore $P(z|t)$ is sufficiently described by two vectors, the mean and variance, which are the outputs of $\mu(\cdot)$ and $\Sigma(\cdot)$ respectively. A detailed description of the architecture used to compute $P(z|t)$:

Embedding layer: Extracts features for each word in the review t_{ui} using a word embedding matrix W, which holds a vector of size k for each word in the vocabulary. We obtain W by pre-training a word2vec model [19] on our data.

Convolutional layer: Gathers locally meaningful information. We apply multiple filters of window sizes $h \in \{3, 4, 5\}$ in order to catch different dependencies between phrases of various lengths and use hyperbolic tangent as the activation function.

Max-over-time pooling operation: For each filter, selects the maximal value across all time windows, as described in [8].

Two feedforward neural networks: Each network is a single fully connected layer, followed by hyperbolic tangent. Their outputs define $P(z|t) = \mathcal{N}(\mu(t), \Sigma(t))$. Note that $\mu(\cdot)$ and $\Sigma(\cdot)$ separate only in the fourth layer.

2.1.1 *Bias Terms.*

In CF techniques, the bias terms may explain the variation in rating values [14]. Inspired by that, when the predicted rating is based on textual reviews, we introduce bias terms for users and items b_u and b_i.

Adding the global mean rating, the conditional probability distribution of the ratings is computed as follows:

$$P(r|u,i,z) = \mathcal{N}(\mu + b_u + b_i + \sum_f g(z)^{[f]},\ \lambda) \tag{3}$$

2.1.2 *Learning.*

We use stochastic gradient ascent to optimize the objective function, as defined in Equation 2. Upon computing the gradient of this equation, the gradient symbol can be pushed into the expectations. This allows us in every step of the optimization, to draw a single quartette $(u, i, r_{ui}, t_{ui}) \in D$ and a single $z \sim P(\cdot|t)$. We would like to iterate that since we optimize via stochastic gradient ascent, a single sampled z may be used as an approximation to the whole distribution. However, we propose to sample from the distribution $P(z|t;\theta)$, which depends on our model parameters. Sampling is a non differentiable operation and would not allow to back-propagate the error through this layer. As a result, the CNN parameters will not be learned. To overcome this barrier, we use the reparameterization trick [13]. Given $P(z|t) = \mathcal{N}(\mu(t), \Sigma(t))$, it is possible to sample from $P(z|t)$ by sampling $\epsilon \sim \mathcal{N}(0, I)$. Then the sampled example is $z = \mu(t) + \Sigma^{1/2}(t) \odot \epsilon$, where $\odot$ denotes an element-wise product. This simple trick moves the sampling to a newly created input layer. Therefore, we modify Equation 2 and optimize:

$$\mathbf{E}_{(u,i,r,t)\sim D}[\mathbf{E}_{\epsilon\sim\mathcal{N}(0,\,I)}[\log P(r|u,i,\mu(t) + \Sigma^{1/2}(t) \odot \epsilon)]]. \tag{4}$$

We emphasize that now the expectations are not with respect to distributions that depend on the model parameters. The entire optimization process is depicted in Figure 1.

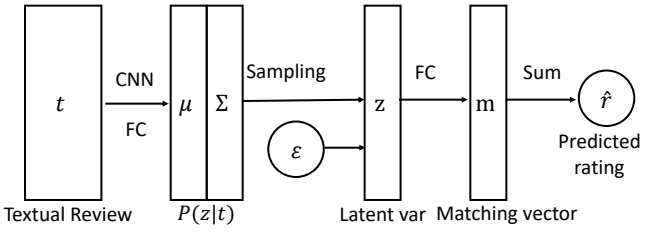

Figure 1: Optimization process

So far we have only discussed the likelihood function. We assume the prior density $P(\theta)$ to be a normal distribution with zero mean and a diagonal covariance matrix. Namely, $P(\theta) = \mathcal{N}(0, \frac{1}{N} * I)$. Integrating the prior and the likelihood into Equation 1, yields the following minimization problem:

$$\sum_{(u,i,r_{ui},\cdot)\in D} \frac{N}{\lambda}(\hat{r}_{ui} - r_{ui})^2 + N\|\theta_1\|^2 \propto \sum_{(u,i,r_{ui},\cdot)\in D} (\hat{r}_{ui} - r_{ui})^2 + \lambda\|\theta_1\|^2 \quad (5)$$

where $\hat{r}_{ui}$ is the predicted rating and θ_1 denotes the parameter set in the first phase.

2.2 Second Phase

The goal of latent factor models is to represent any user u and item i with an f-dimensional vector $p_u, q_i \in \mathbb{R}^f$, such that their inner product indicates their affinity. Formally, the predicted rating is computed as $\hat{r}_{ui} = p_u^T \cdot q_i + b_u + b_i + \mu$. In essence, this phase can integrate any underlying collaborative filtering algorithm, but now the textual reviews are digested, and the matching vectors m_{ui} are used instead. Namely, the input is $D' = \{(u, i, r_{ui}, m_{ui})\}$. To obtain m_{ui}, for each review t_{ui} we first apply the first phase and infer the distribution $P(z_{ui}|t_{ui})$. We then set z_{ui} as the mode of the distribution ($z_{ui} = \mu(t_{ui})$) and compute $c = g(z; \theta)$.

While $p_u^T \cdot q_i = \sum_f p_u^{[f]} q_i^{[f]} = \sum_f (p_u \odot q_i)^{[f]}$ indicates the overall matching between user u and item i, we take a higher granularity and observe the element-wise product $p_u \odot q_i$. Since now the label set includes the matching vectors m_{ui}, the loss function may consider individually each of the f values of the element-wise product between the user and the item $p_u \odot q_i$, rather than only their sum $p_u^T \cdot q_i$. It is inevitable that on top of the scalar labels (r_{ui}), the multi-faceted labels (m_{ui}) would give additional insights about the opinion of the user towards the item. This approach naturally improves the optimization process. If a scalar label is used, then the partial derivatives the user and items vector will be such that *all* f element-wise products will be increased/decreased to minimize the loss. Therefore the latent factors are *coupled*, and possibly a better solution, that increases some of the element-wise products and reduces the rest, is beyond the search space. In contrast, our proposed algorithm obtains an independent label per each dimension. This in turn *decouples* the latent factors and helps to converge to a better solution.

Inspired by that, the loss function of the second phase is a convex combination of the loss function of the underlying CF algorithm and the prediction error of the matching vector. To highlight the effectiveness of our approach, we choose simple SVD [20] as the CF method, with predicted rating $\hat{r}_{ui} = p_u^T \cdot q_i + b_u + b_i + \mu$. The loss function is:

$$\sum_{(u,i,r_{ui},m_{ui})\in D'} (\hat{r}_{ui} - r_{ui})^2 + \alpha \cdot \|p_u \odot q_i - m_{ui}\|_2^2 + \lambda_2 \|\theta_2\|^2 \quad (6)$$

where λ_2 is the regularization factor, θ_2 is the parameter set of the second phase and α controls the relative importance of terms in the loss function.

3 RELATED WORK

Most recommender systems have been designed to handle structured input [24], and cannot incorporate textual reviews. The majority of algorithms that can handle textual data employ topic modeling to represent users and items. For example, In [2, 17], LDA [3] was applied on the reviews, resulting in a significant improvement over traditional SVD methods. In [9] a probabilistic model based on collaborative filtering and topic modeling was proposed. This modeling uncovers aspects and sentiments of users and items, but it does not incorporate the explicit ratings during modeling reviews.

[6] introduce LRPPMCF, a tensor matrix factorization algorithm to Learn to Rank user Preferences based on Phrase-level sentiment analysis across multiple categories, combined with a CF method. [21] presented the algorithm Aspect-based Latent Factor Model (ALFM), that combines ratings and review texts to improve rating predictions. [12] automatically extracts aspects from the reviews to generate user–item–aspect ternary relations. [29] applies LDA on the reviews and used Tensor Factorization to model users and items. [30] introduced the Explicit Factor Model (EFM) to generate explainable recommendations according to the specific product aspects. [15] presents the Ratings Meet Reviews (RMR) algorithm, that applies topic modeling techniques on item review text and aligns the topics with the users dimensions to improve prediction accuracy. However, the user-item rating prediction is based on the probability the words a user usually writes also appear in the words written on the item. Hence, the rating prediction does not necessarily encodes "likeness". Additionally, in order to align the dimensions, they cannot use MF and use mixture of Gaussians instead. [1] proposed an algorithm that does not model users and items explicitly in a joint manner from their reviews, but uses reviews to regularize their model. Collaborative Deep Learning (CDL) [28] jointly performs collaborative filtering for the explicit ratings and deep representation learning for the content information.

These work lack natural language understanding, and do not capture semantic meaning. Deep learning methods were recently introduced to overcome these limitations: Deep Cooperative Neural Networks (`DeepCoNN`) was proposed in [31], which trains two separate sub-networks. One to model the user by processing the concatenation of all the reviews written by this user. Similarly, another network models the item using the concatenation of the textual reviews written about this item. Then the user and item representations are combined using a factorization machine [22] to predict the rating. During training, while predicting the rating of user A on item B, the corresponding joint textual review is accessible to algorithm. During testing, the joint review cannot be revealed, which creates inconsistency. A variant named `DeepCoNN-revAB` copes with the aforementioned inconsistency by excluding the joint reviews from the training set.

[4] presented `TransNets`, which improves `DeepCoNN`. They introduced a third sub-network, which models the target user-item pair. They further presented an extension, `TransNets-Ext` which also contains user and item embedding matrices. These algorithms were evaluated on the most recent publicly available datasets and obtained state-of-the-art results. Their relative advantage is in the cold user scenario, a prevalent problem in recommenders [23]. An analysis of the results shows the superiority of the extended version,

	SVD	DeepCoNN	DeepCoNN-rev$_{AB}$	TransNet	TransNet-Ext	MDR	Improved
Yelp	1.8661	1.8984	1.7045	1.6387	1.5913	**1.4257**	10.4%
A-Electronics	1.8898	1.9704	2.0774	1.8380	1.7781	**1.5329**	13.8%
A-Clothes	1.5212	1.5487	1.7044	1.4487	1.4780	**1.2837**	13.1%
A-Movies	1.4324	1.3611	1.5276	1.3599	1.2691	**1.1782**	7.2%

Table 1: MSE Comparison with baselines. Best results are indicated in bold.

	Reviews	#users	#items
Yelp	4,153,150	1,029,432	144,072
A-Electronics	7,824,482	4,200,520	475,910
A-Clothes	5,748,260	3,116,944	1,135,948
A-Movies	4,606,671	2,088,428	200,915

Table 2: Datasets statistics

except for a single case with an extreme sparse usage matrix. [25] can be considered as a sub-network of TransNets-Ext.

[27] proposed a CNN based model identical to `DeepCoNN`, but with an attention mechanism to construct latent representations of users and items. Later [5] designed another variant of `DeepCoNN`, which also uses an attention, in order to asses the importance of each review while modeling either users or items. [7] suggest a different variant of `DeepCoNN` using attention mechanism, to learn the importance of various latent features to users. [16] present an algorithm that uses reviews as priors for user and item vectors. However, since their algorithm invokes matrix inversion operation, its running time is proportional to $O(k^3)$, where k is the latent space dimensionality, comparing to linear dependency in k in our proposed method. A common drawback to all these solutions is scalability issues. Processing each review requires feeding to the network a concatenation of all reviews associated with the user and the item. To demonstrate it, let us consider a simple case where each user has 10 reviews and each item has 100 reviews. Current methods will go over each review 110 times, while our proposed algorithm makes a single pass on each review. To mitigate this problem [26, 27] evaluated their algorithm on relatively small datasets. But such naive solution emphasizes the importance of designing an efficient algorithm, like ours, that can handle large datasets.

4 RESULTS

4.1 Datasets

In our experiments, we used four well-known datasets to evaluate our model. Each user-item interaction in these datasets contains both a textual review and an explicit rating. They cover different domains and vary in their review writing styles, average length of reviews and usage patterns. Therefore, they allow to evaluate the robustness of our proposed approach. The first one `Yelp17`, is a large-scale dataset consists of restaurant reviews, introduced in the recent Yelp Challenge [1].The other datasets are three of the larger datasets in the latest release of Amazon reviews [2] [18]. These datasets contain product reviews from Amazon website over the period of May 1996 - July 2014. We use the de-duplicated version of the dataset and also discard entries where the review text is empty. Statistics of the datasets are summarized in Table 2.

[1] https://www.yelp.com/dataset-challenge
[2] http://jmcauley.ucsd.edu/data/amazon

4.2 Baselines and Evaluation Metric

In order to validate the superiority of our approach, we have used 5 baselines, where their descriptions are detailed in Section 3: `SVD` (A purely rating based model [14]), `DeepCoNN`, `DeepCoNN-revAB`, `TransNet` and `TransNet-Ext`.

Since we evaluate on common benchmark datasets, we did not implement these baselines. Instead, we report the results of these baselines as reported in the state-of-the-art [4] with the exact same data settings. In that paper, randomly selected (80%, 10%, 10%) of the users are used for the training, validation and test sets. All baselines were implemented (where two of our baselines were proposed in that paper), and hyper parameters were tuned on a validation set using grid search.

Throughout the experiments we measure the performance of the algorithms using Mean Square Error (MSE)[11]. It is selected because the recent related work and the strongest available baselines have used the same evaluation metric [1, 4, 15, 17, 31]. As future work, we plan to evaluate on precision metrics as well.

4.3 Setup and Method implementation

We implemented the proposed algorithm using TensorFlow [3] employed with the AdaGrad optimizer [10]. For all of our experiments the first phase was trained with 128 filters of each size: 3,4 and 5. The size of the matching vectors was chosen to be 50, equal to the size of the user and item vectors. While training the first phase, we applied batch normalization and used dropout with 0.5 probability on both the convolutional layer and the fully connected layer.

In the next phase, the weight of α was chosen to be between $[0.0, 1.5]$, where a value of 0 means that the matching vectors are completely not indicative for the rating prediction task. Across all datasets, we found that the optimal value was close to 1, which suggests the importance of user reviews. λ_1 and λ_2 were chosen from the range $[0.1, 0.5]$.

4.4 Comparison with Baselines

Table 1 reports the main results of our experiments on the four evaluated datasets. The rightmost column presents the improvement in performance achieved by `MDR` comparing to the best baseline. Analyzing the baseline results confirms that using reviews improves performance, as `SVD` was inferior to the others. It is clearly shown that `MDR` stands out from the rest, gains a 10.4%, 13.8%, 13.1% and 7.2% improvement comparing to the strongest baseline in the Yelp, Amazon-Electronics, Amazon-Clothes and Amazon-Movies datasets respectively. This major lift in performance proves that the intuition of the algorithm, together with a deep learning architecture to capture the essence of user generated reviews, better models users and items.

[3] https://www.tensorflow.org/

REFERENCES

[1] Amjad Almahairi, Kyle Kastner, Kyunghyun Cho, and Aaron Courville. 2015. Learning distributed representations from reviews for collaborative filtering. *RecSys* (2015).

[2] Yang Bao, Hui Fang, and Jie Zhang. 2014. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation.. In *AAAI*, Vol. 14. 2–8.

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

[4] Rose Catherine and William Cohen. 2017. TransNets: Learning to Transform for Recommendation. *RecSys* (2017).

[5] Chong Chen, Min Zhang, Yiqun Liu, and Shaoping Ma. 2018. Neural Attentional Rating Regression with Review-level Explanations. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 1583–1592.

[6] Xu Chen, Zheng Qin, Yongfeng Zhang, and Tao Xu. 2016. Learning to rank features for recommendation over multiple categories. In *SIGIR*. ACM.

[7] Zhiyong Cheng, Ying Ding, Xiangnan He, Lei Zhu, Xuemeng Song, and Mohan S Kankanhalli. 2018. Aˆ 3NCF: An Adaptive Aspect Attention Model for Rating Prediction.. In *IJCAI*. 3748–3754.

[8] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12 (2011), 2493–2537.

[9] 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 *KDD*. ACM.

[10] John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research* 12, Jul (2011), 2121–2159.

[11] Asela Gunawardana and Guy Shani. 2009. A survey of accuracy evaluation metrics of recommendation tasks. *Journal of Machine Learning Research* 10, Dec (2009), 2935–2962.

[12] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *CIKM*. ACM.

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

[14] Yehuda Koren, Robert Bell, and Chris Volinsky. 2009. Matrix Factorization Techniques for Recommender Systems. *Computer* 42, 8 (Aug. 2009), 30–37. DOI: http://dx.doi.org/10.1109/MC.2009.263

[15] Guang Ling, Michael R Lyu, and Irwin King. 2014. Ratings meet reviews, a combined approach to recommend. In *RecSys*. ACM.

[16] Yichao Lu, Ruihai Dong, and Barry Smyth. 2018. Coevolutionary Recommendation Model: Mutual Learning between Ratings and Reviews. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 773–782.

[17] Julian McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. ACM.

[18] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *CIKM*. ACM.

[19] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

[20] Arkadiusz Paterek. 2007. Improving regularized singular value decomposition for collaborative filtering. In *KDD cup and workshop*, Vol. 2007.

[21] Lin Qiu, Sheng Gao, Wenlong Cheng, and Jun Guo. 2016. Aspect-based latent factor model by integrating ratings and reviews for recommender system. *Knowledge-Based Systems* 110 (2016), 233–243.

[22] Steffen Rendle. 2010. Factorization machines. In *ICDM*. IEEE, 995–1000.

[23] Oren Sar Shalom, Shlomo Berkovsky, Royi Ronen, Elad Ziklik, and Amir Amihood. 2015. Data Quality Matters in Recommender Systems. In *Proceedings of the 9th ACM Conference on Recommender Systems*. ACM, 257–260.

[24] Oren Sar Shalom, Haggai Roitman, Amihood Amir, and Alexandros Karatzoglou. 2018. Collaborative Filtering Method for Handling Diverse and Repetitive User-Item Interactions. In *Proceedings of the 29th on Hypertext and Social Media*. ACM, 43–51.

[25] Oren Sar Shalom, Guy Uziel, Alexandros Karatzoglou, and Amir Kantor. 2018. A Word is Worth a Thousand Ratings: Augmenting Ratings using Reviews for Collaborative Filtering. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*. ACM, 11–18.

[26] 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.

[27] Sungyong Seo, Jing Huang, Hao Yang, and Yan Liu. 2017. Representation Learning of Users and Items for Review Rating Prediction Using Attention-based Convolutional Neural Network. In *International Workshop on Machine Learning Methods for Recommender Systems*.

[28] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative deep learning for recommender systems. In *KDD*. ACM.

[29] Chong Yang, Xiaohui Yu, Yang Liu, Yanping Nie, and Yuanhong Wang. 2016. Collaborative filtering with weighted opinion aspects. *Neurocomputing* 210 (2016), 185–196.

[30] 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 *ACM SIGIR*.

[31] Lei Zheng, Vahid Noroozi, and Philip S Yu. 2017. Joint deep modeling of users and items using reviews for recommendation. In *ACM WSDM*.