User Feedback-Based Rating Prediction in Recommendation System

—— 2023 SDSC4016 Final Project

Team 9

WU XIAOQING 56641363
LI YISHAN 56641302

CONTENT

1. Motivation

With the increasing demand for online shopping, rating prediction under users' preferences has been a hot research topic in the e-commerce recommendation domain. While social commerce is developing rapidly, customer feedback, especially various customer reviews and product ratings, has become more influential for companies to understand customer preferences. More emotional and semantic messages are delivered in the review text. Firstly, the review content specifically contains the reason for their ratings accordingly. For example, 3-star and 4-star reviews usually describe the product's amazing features, along with where the reviewers expect an improvement. Secondly, a customer might rate and give different reviews on several similar products, which will help to understand customers' preferences deeply, with enhanced rating prediction accuracy for further recommendation (Zhai and Peng, 2016).

A precise rating prediction enables recommending items under users' preference, therefore providing a more personalized experience that meets customers' satisfaction. Moreover, precise rating prediction that understands customer trends can help improve profits, sales, and customer loyalty, enhancing enterprises' core competitiveness (Schafer et al., 2001, Grbovic et al., 2015).

2. Background

A large range of deep learning methods has been applied to rating prediction for recommendation systems. Deep learning methods can currently be applied well in rating prediction, especially when incorporating review texts and rating matrix factorization. Word vector methods are commonly applied by mapping the sequence relationship between target and context words and training word vectors obtained by a neural network, where word2vec and GloVe (Global Vectors for Word Representation) are the most common methods (Jeffrey et al., 2014). Researchers have presented a convolutional matrix factorization that incorporates CNN into probabilistic matrix factorization (Kim et al., 2016). Chen et al. (2018) applied the review text and rating matrix model inputs to improve the rating prediction performance. Negin (2021) improved recommendation systems by computing users' similarity by utilizing several different approaches, where the model with Long Short Term Memory (LSTM) outperformed other models.

Moreover, recommendation technologies based on rating matrices are also a common research topic in previous works. It generally has two categories: neighborhood-based (predict ratings based on similar users or items) and model-based algorithms (Zhu et al., 2019). Generally, neighborhood-based methods which have more intuitive explanations, however, also suffer from weaknesses. The first would be data sparsity, which implies data is not enough to compute the similarity between users. Also, usually there is insufficient data to recommend items for new users, which is another weakness, cold start issues. To overcome those problems, researchers proposed latent factor models (Forsati et al. 2014), among which matrix factorization-based models (MF) have received great focus (Bokde et al. 2015). MF methods transform recommendation tasks into rating-matrix completion problems by MF operation and are proven effective. Currently, many research studies incorporate MF methods with deep neural network models based on review text and rating data for rating prediction.

3. Description

We aim to establish a predictive model of how each user was most likely to rate each item based on known information. And we divided it into 2 parts based on information type,

user-item network, and review text. To achieve a better result, we use BERT pre-trained model to process the user review information and the Surprise library to create a user-item network.

3.1 BERT: Bidirectional Encoder Representations from Transformers

BERT, which stands for Bidirectional Encoder Representations from Transformers, is a language representation model published by researchers in Google AI Language. BERT is conceptually simple and empirically effective, obtaining a new state-of-art performance on NLP tasks. Therefore, we aim to establish a prediction model based on BERT in our project.

There are two steps in the framework of BERT: 1) pre-training: training the model on unlabeled data from different tasks; 2) fine-tuning: first initializing the BERT model with pre-trained parameters, then fine-tuning all parameters using labeled data over downstream tasks. The model architecture of BERT is a multi-layer bidirectional Transformer encoder based on Transformer. It is designed as pre-trained bidirectional representations of unlabeled content. The model learns to jointly predict both contexts on the left and right sides in all layers, which is so-called bidirectional. As shown in Figure 1, BERT applies a bidirectional Transformer while OpenAI GPT applies a left-to-right Transformer and ELMo uses independently trained right-to-left and left-to-right LSTMs.

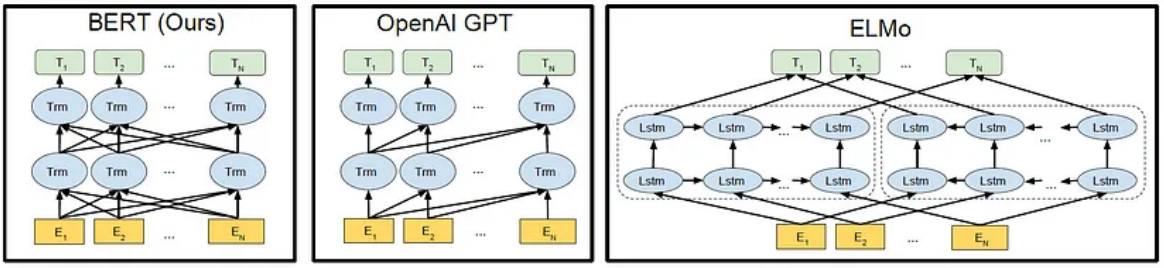

Figure 1: BERT using bidirectional Transformer

BERT is pre-trained using two unsupervised training strategies: MaskedLM (MLM) and Next Sentence Prediction (NSP). MLM is the procedure to train a deep bidirectional model by randomly masking a specific percentage of the input tokens and then predicting the masked tokens. Specifically, researchers randomly mask 15% of the WordPiece tokens in every sequence. Instead of reconstructing all of the inputs, they only predict the masked words. When choosing the i-th token A, 80% of the time the chosen token is replaced with the masked token, and 10% of the time with a random token, 10% of the time with the unchanged chosen token. Then the original token is predicted with cross-entropy loss. NSP is a binarized task to obtain a pre-trained model that understands relationships between sentences. Specifically, if two sentences A and B are chosen for each example for pretraining, 50% of the time it is labeled as IsNext (B is actually the next sentence of A) and labeled as NotNext 50% of the time (B is a random sentence of the corpus).

The fine-tuning step is straightforward. We plug in specific inputs and outputs into the pre-trained model for specific language tasks, then fine-tune parameters end-to-end. For instance, for sentiment classification tasks, we need to add a classifier layer on the top of the Transformer output for the [CLS] token, where the [CLS] token is a symbol that is added in front of each input example. Figure 2 shows an overall procedure of the BERT model, using Question-answering tasks as an example.

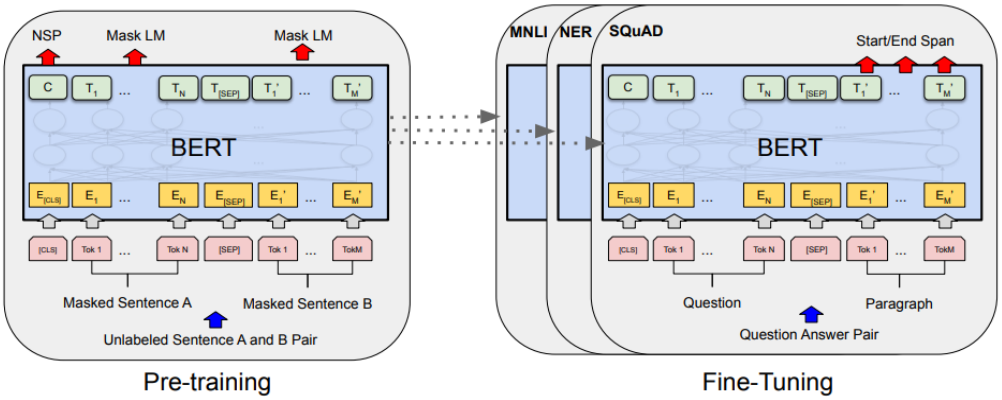

Figure 2: an overall procedure of BERT: pre-training and fine-tuning steps

3.2 Processing user-item network based on rating matrices

We used a Python library called Surprise to process the used-item network based on rating matrice. Surprise is a Scikit-based library to develop recommendation systems, which contains rich prediction algorithms such as baseline algorithms, neighborhood methods, matrix factorization-based (NMF, SVD, PMF, SCD++, etc.), and many others. The prediction methods we applied for rating prediction are listed following with a brief description:

NMF. Non-negative matrix factorization is an unsupervised collaborative filtering algorithm. It projects data into a lower dimension, effectively decreasing the features with retained necessary information for the reconstruction of the original data (Lee & Seung, 2001).

SVD. Singular value decomposition is a powerful and very popular linear algebra technique of dimensionality reduction, breaking down a matrix into the product of smaller matrices. It also has been commonly used to build a recommendation system. Furthermore, an optimized SVD technique (SVD++) generates implicit information for more accurate prediction.

PMF. A classical probability matrix factorization model based on SVD, with further adopting the latent factor vector representing user preferences and item features with Gaussian prior distribution.

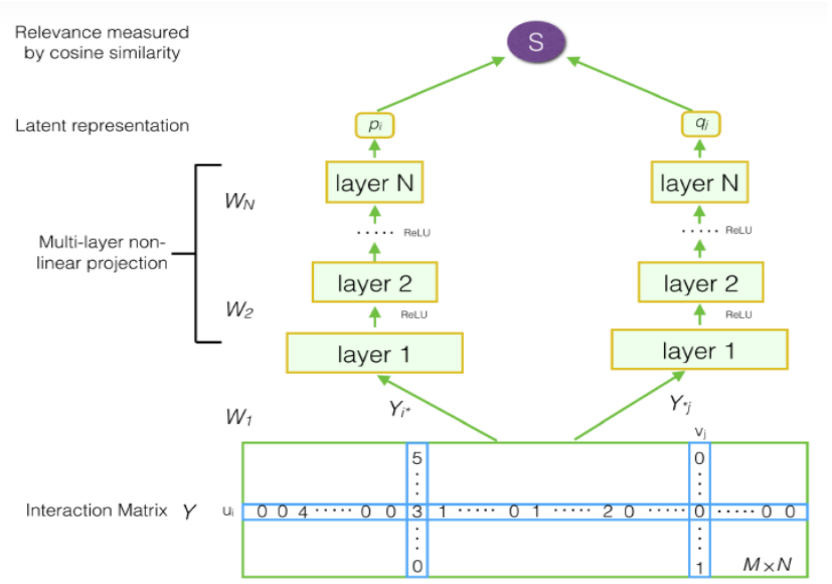

Figure 3: the model architecture of DMF

For further recommendation, we also applied a deep matrix factorization model (DMF) with a neural network architecture that maps the items and users into a common low-dimensional space by non-linear projections. It has been proven to outperform other state-of-art recommendation methods. Figure 3 illustrates the model architecture of DMF. Lastly, a designed loss function is introduced for optimization, followed by the training algorithm.

4. Data

We conducted experiments on the public data sets from Amazon (2018), containing the reviews and metadata with 34.6 thousand reviews from May 1996 to July 2014. Three types of data are selected: review text, users, items, and ratings. The 5-core dataset, where each user or item has at least 5 reviews, is chosen because more information is contained by continuous users than by single ones. Table 1 shows the key statistics of the dataset. Each review has one rating data accordingly.

Amazon review dataset

Number of reviews	34625	Avg. of ratings	4.2862
Number of items	1581	Std. of ratings	1.0374
Number of users	3818	# of ratings per item	21.6730
# of user-item links	28064	# of ratings per user	8.9746

Table 1: key dataset statistics

5. Implementation

We divided this rating prediction system into 2 parts according to the data type it utilized. After data preprocessing, the text part would be embedded to fix the transformer and LSTM; the user-item network would be loaded into Surprise and trained with numerous models to find the optimal one. Here is the schematic diagram which elaborates the training process.

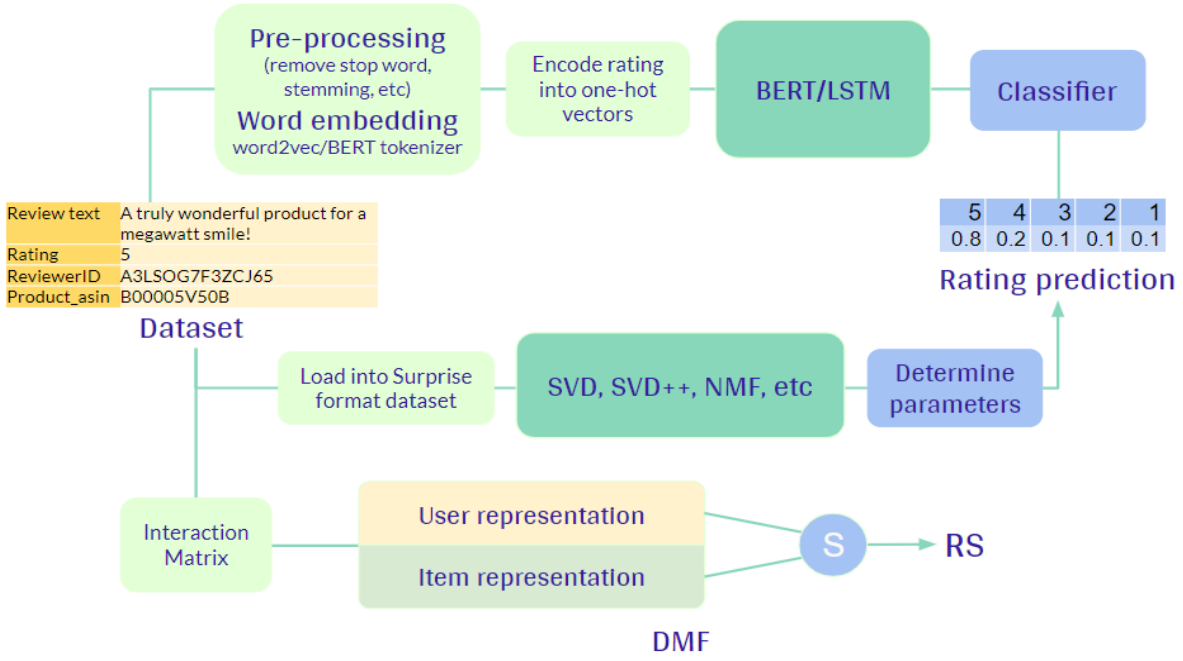

Figure 4: schematic diagram of the project implementation

5.1 Rating Prediction on User-Item Network

- Load dataset into surprise format dataset
- Test with cross-validation method and find out the best model
- Use Grid search to determine the best parameters according to RMSE
- Train model with upper information
- Analysis and visualize the results

To help you obtain an easier and more fluent reading experience of our training code, we list the variables we assigned with their meanings. Besides, the optimal hyperparameters after several tryings are also decided. The followings are the variables and parameters that appeared in the code of the user-item network part:

reader: to decide how we load the data file into the Surprise recommendation model. We specified the rating range as (1,5)
data: the dataset with information of userID, itemID and rating which is loaded in the format of Surprise Datasets.
algo: the algorithm used to predict the relationship between rating and user-item information
perf: the information on the training process, including training time and the loss according to the selected method.
param_grid: the parameters range that we want to test
best_params: the best model that we obtain from the grid search
trainset, test set: divided from the prime dataset (data), with the ratio of 4:1
percentage_correct: the percentage of correct prediction
test_mse: the list for testing RMSE of every epoch.
epoch: 500

For further recommendation, we also applied a deep matrix factorization model (DMF) with a neural network architecture that maps the items and users into a common low-dimensional space by non-linear projections.

- Preprocess data, generate training/test set, and load them into dataloader
- Built initial embedding vectors of users/items
- Train the DMF model with neural network (user representation and item representation)
- Compute NDCG (a measure considering the position of relevant items in the ranked list) and HR (hitting rate)
- Analysis and visualize the results

To help you obtain an easier and more fluent reading experience of our training code, we list the variables we assigned with their meanings. Besides, the optimal hyperparameters after several tryings are also decided. The followings are the variables and parameters that appeared in the code of the DMF part:

ui_data: a dataframe containing userID, itemID, and ratings in integer format.
getTrainTest: a function to split the training set and test set. The first n-1 of each user is the training set, and the last one is the test set.
generate_train_dataset: a function for sampling on the training set
generate_test_dataset: a function for negative sampling on test set
negative_num: number of negative sampling on training set/test set
DMF: the established model where we first initialize user/item embedding vectors, then put them into the multi-layer neural network and lastly relevance measure by cosine similarity.

generate_useritem_matrix: return a user-item matrix
maxRate: set 5
optimizer: Adam optimizer is applied
getNDCG: a measure of the effectiveness of a recommendation system, taking into account the position of relevant items in the ranked list. Record when a target item exists in the top K rank list.
getHR: Hitting rate. Hit if a target item is in the predicted top K rank list, return 1 (otherwise return 0)
Target: target items in the test set
Ranklist: predicted score sorting of 1 real target item for each user and 99 negatively sampling items

5.2 Rating Prediction on Review Text

- Load dataset and pre-processing (lowercase, remove stop word and negation, split individual words, stemming)
- Word embedding by using word2vec or BERT tokenizer
- Encode rating into one-hot vectors
- Fit data into Datasets and load into data loader
- Establish the model (LSTM/BERT)
- Train model and output results

To help you obtain an easier and more fluent reading experience of our training code, we list the variables we assigned with their meanings. Besides, the optimal hyperparameters after several tryings are also decided. The followings are the variables and parameters that appeared in the code of review text part:

df: Amazon Review dataset with the format of Pandas.DataFrame
identify_tokens: a function that splits the sentences of every row into individual words, returns the word list
remove_stops: a function that removes the stop word from the word list of each row and returns a modified word list
remove_negation: a function that removes the negation like 'n't' from the word list and substitutes it with "not", returning the modified word list
stem_list: a function that concludes the keywords of the word list by Portstemmer, returns the keyword list
rating_encode: a function that encodes rating into the 5-dimension one-hot vector, returns the one-hot vector
sen_len: the maximum length of each input sentence
w2v_path: the address where to load Word2Vec embedding
model: the deep learning algorithm, we use LSTM and BERT
optimizer: apply Adam optimizer
criterion: cross-entropy loss
scheduler: to control the learning rate using ReduceLROnPlateau
max_acc: keep the best record of test accuracy
tokenizer: the word embedding method from BERT
input_ids: numerical vector converted from words
attention_mask: mark the padding of each sentence
flat_accuracy: function to compare predictions and targets and output the prediction accuracy

6. Results and Observations

After training the dataset with those models, the accuracy of rating prediction from different models is as follows. The prediction accuracy of the user-item network is 61.537%; and for the review text, the LSTM model's prediction accuracy is approximately 43.868%, while the prediction accuracy of the BERT model reaches 56.858%。

Here are some observations that we analyzed from the results. According to Figure 5, we can see that compared to the LSTM model, the transformer encoder model BERT has lower loss and higher accuracy, which indicates that the transformer model is more well-performing in the aspect text sentiment analysis than the LSTM model.

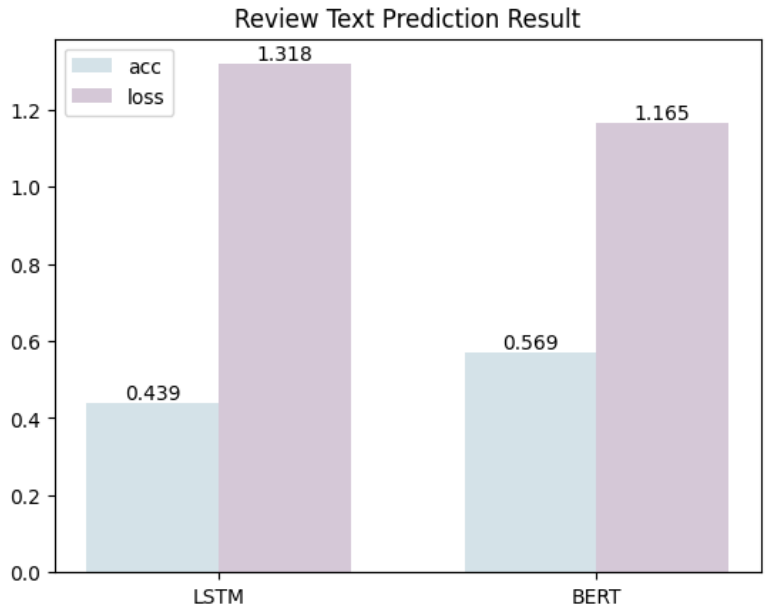

Figure 5: the accuracy and loss of LSTM and transformer(BERT)

Referring to Figure 6, dynamic RMSE with training shows that in the early stage of training the RMSE decreases fast while the decreasing speed becomes slow with epochs increasing. At about the 80 epoch, the RMSE reached a stable platform. This curve is a typical RMSE change during the training. Also, in the evaluation plot of DMF shown in Figure 6, the hitting rate comes to 54%, and NDCG comes to 40% with a stable changing stage.

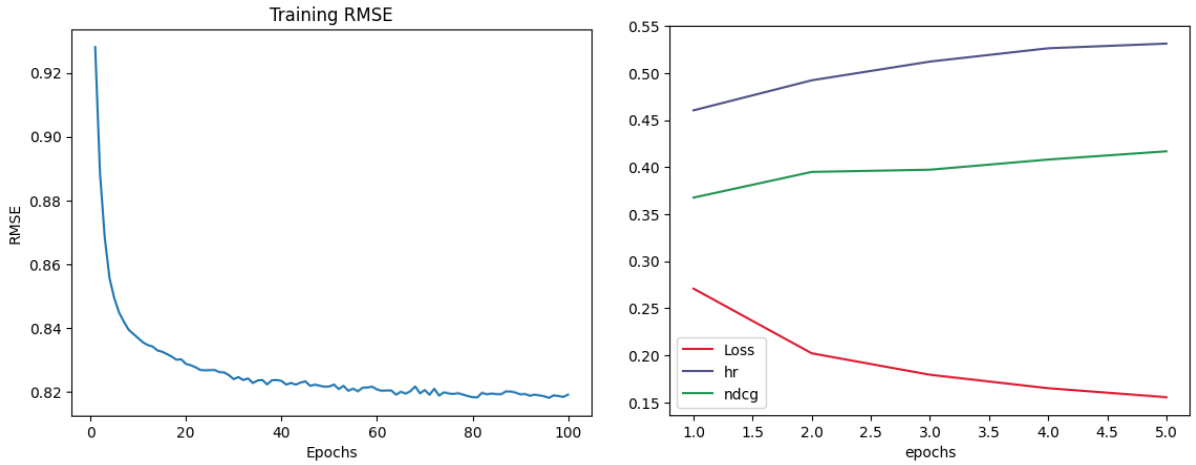

Figure 6: the change of evaluations of each MF model with epoch increasing

7. Discussion

7.1 Limitations

Apart from the model elaborated above, we have actually tried other deep learning models like GNN, etc. But neither of them achieve satisfactory results. In a word, it seems that deep learning models show poor performance on rating prediction of the Amazon Review Dataset. We assume the reasons are the following:

1) Biased Dataset. We could see that more than half of the rating is 5, and the portion that reviews ratings of 2 or 3 is extremely small. This situation results in the deep learning model keeping focusing on the full rating and ignoring some features of rare rating review text.

2) One-hot vector encoding. The way we choose to encode the label does not totally reflect its feature. As this is a five-class classification, we transform the rating into 5-dimension one-hot vector labels. But this method ignores the level or sequence of different ratings. To elaborate, a rating of a higher number means a more positive attitude in review text, while the one-hot encoder shows all classes are equal. And we also take the tutor's advice to turn this 5-class problem into a 2-class by categorizing 3,4,5 rating scores to 1 the others are 0. However, the final accuracy discloses a suffocating and unbelieving decline compared with the one-hot encoding method.

3) Device defect. Due to the limitation of GPU memory, we gave up establishing items to items network using metadata. We intend to use such a network to help us find out the similarity between commodities that may point to a similar rating. While we were stuck at encoding them into one-hot vectors as the number of commodities reached millions.

7.2 Future Prospect

In the future, we are determined to find some methods to relieve the bias of the model. A feasible solution is using a focal loss function which can force the model to focus on the minor classes. Compared with cross-entropy loss, the focal loss did not change for samples with inaccurate classification but decreased for those with accurate classification. On the whole, it is equivalent to increasing the weight of inaccurate samples in the loss function.

On the other side, we intend to optimize the model structure. The current model we used reveals the relationship between the rating and user-item network as well as review texts. In fact, the correlation of items themselves also takes place. For example, items that are purchased together usually have similar ratings. We plan to find a model that can utilize the items network with association rule mining in order to establish a better prediction system.

Also, several weaknesses have been revealed in our project. The recommended performance of DMF is not ideal and much worse than the experiment in the original DMF paper. Besides the biased dataset and device defect limitations, we should verify the DMF model with a pair-wise objective function rather than point-wise. The scalability of DMF should also be further verified.

Reference:

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., … Polosukhin, I. (2017). Attention Is All You Need. https://doi.org/10.48550/arxiv.1706.03762

Xue, H., Dai, X., Zhang, J., Huang, S., & Chen, J. (2017). Deep Matrix Factorization Models for Recommender Systems. International Joint Conference on Artificial Intelligence. https://doi.org/10.24963/ijcai.2017/447

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. https://doi.org/10.48550/arxiv.1810.04805

Lü, L., Medo, M., Yeung, C. H., Zhang, Y.-C., Zhang, Z.-K., & Zhou, T. (2012). Recommender systems. Physics Reports, 519(1), 1–49. https://doi.org/10.1016/j.physrep.2012.02.006

Forsati, R., Mahdavi, M., Shamsfard, M., & Sarwat, M. (2014). Matrix Factorization with Explicit Trust and Distrust Side Information for Improved Social Recommendation. ACM Transactions on Information Systems, 32(4), 1–38. https://doi.org/10.1145/2641564

Salakhutdinov, R., & Mnih, A. (2008). Bayesian probabilistic matrix factorization using Markov chain Monte Carlo. Proceedings of the 25th International Conference on Machine Learning, 880–887. https://doi.org/10.1145/1390156.1390267

Cichocki, A., Zdunek, R., & Amari, S. (2006). Csiszár's Divergences for Non-negative Matrix Factorization: Family of New Algorithms. Independent Component Analysis and Blind Signal Separation, 32–39. https://doi.org/10.1007/11679363_5

Liu, W., Wu, C., Feng, B., & Liu, J. (2015). Conditional preference in recommender systems. Expert Systems with Applications, 42(2), 774–788. https://doi.org/10.1016/j.eswa.2014.08.044

Coussement, K., De Bock, K. W., & Geuens, S. (2022). A decision-analytic framework for interpretable recommendation systems with multiple input data sources: a case study for a European e-tailer. Annals of Operations Research, 315(2), 671–694. https://doi.org/10.1007/s10479-021-03979-4

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix Factorization Techniques for Recommender Systems. Computer (Long Beach, Calif.), 42(8), 30–37. https://doi.org/10.1109/MC.2009.263

Kim, D., Park, C., Oh, J., Lee, S., & Yu, H. (2016). Convolutional Matrix Factorization for Document Context-Aware Recommendation. Proceedings of the 10th ACM Conference on Recommender Systems, 233–240. https://doi.org/10.1145/2959100.2959165

Ghasemi, N., & Momtazi, S. (2021). Neural text similarity of user reviews for improving collaborative filtering recommender systems. Electronic Commerce Research and Applications, 45, 101019–. https://doi.org/10.1016/j.elerap.2020.101019