Recommendation Approach Based on TwinBERT Model

Zheng Luo *

College of Information Science and Technology, Jinan University, Guangzhou, 511436, China

* Corresponding Author Email: LDDDDD@stu2019.jnu.edu.cn

Abstract. In order to provide more personalized recommendation services for users as well as improve the platform's traffic and user satisfaction, a recommendation method based on the TwinBERT model has been proposed. As a matter of fact, this method combines not only the basic information of users and items but also user comments or item text information to predict users' behaviors and make more accurate recommendations. The text information in the dataset is encoded in sentences using the BERT model, and the encoded text features are fused with other features to obtain a representation of user and item embedding vectors. These vectors are then used to predict the user's rating of the item and recommend relevant items for the user. The proposed method was tested on the Amazon review dataset and the MovieLens dataset, and the experimental results showed that it improved over traditional recommendation algorithms in terms of accuracy and efficiency.

Keywords: TwinBERT model, recommendation services, dataset, user comments, text information.

1. Introduction

As the Internet continues to produce massive amounts of data, artificial intelligence technology advances and becomes more popular, and living standards improve, people's desire for online entertainment grows. This has led to the emergence of various online services. In today's world, where there is an overwhelming amount of information available, it can be quite challenging for information producers to provide the relevant information to the right user. Similarly, users can find it difficult to locate the products or information they are looking for amidst a vast sea of data. Therefore, personalized recommendation systems that can solve the information overload problem to a certain extent and generate personalized recommendations according to user preferences have emerged and have been widely used in major platforms. Traditional recommendation methods are mainly based on user behavior data and item metadata, and with the rise of social media and the emergence of a large number of Internet text data, recommendation methods based on text information have also become one of the research hotspots. Meanwhile, the continuous development of natural language processing technology also provides more support for recommendation methods based on text information.

Traditional information retrieval and recommendation systems are no longer sufficient to meet the needs of users, particularly in the realms of e-commerce, social media, and online news, where users need to quickly and accurately find the information they are interested in, and textual information contains a multitude of hidden features. In recent years, deep learning-based natural language processing has made significant strides in the fields of information retrieval and recommendation. For platform parties:

- It can efficiently utilize the traffic to increase the exposure of the corresponding items and help users find what they like, thus increasing the probability of transactions on the platform.
- Maximizing user attraction, retention, and conversion rate leads to achieving business objectives on the platform.
- Regardless of whether the platform is directly profitable or not, the ability of the platform to match the content with the corresponding users is an important criterion for measuring the level of the platform, and the recommender system provides the platform with the ability to match demand and supply.

For items, it can create more exposure opportunities for long-tail items [1]. Within the platform, there is a huge number of items, there is a long-tail nature and the principle of two-eight [2], but

popular items can only cover the needs of a very small number of people, and more needs are still to be obtained from the long-tail items. For users, it is available of matching user demand and supply can improve the user experience. For a user whose needs are clear, the search can directly meet the user's needs; for a user whose needs are not clear, the recommendation can obtain additional surprises and improve the user experience. According to their strategy perspective, traditional recommender system algorithms can be classified into content-based, collaborative filtering and hybrid recommendation approaches. The following presents the current state of research on these recommendation models.

- Content-based recommendation algorithms [3] were the first to be used, relying on the user's history of preferred items to suggest similar ones. Content-based recommendation algorithms involve three key steps: content characterization, feature learning, and generating recommendation lists. This type of algorithm is particularly effective at suggesting relevant but lesser-known items to the user. Thanks to the unique characteristics of each item being taken into account, the recommendations generated are highly accurate. However, the model generalization ability of the content-based recommendation method [4] is weak, and it is easy to fall into the state of "overfitting" [5], and it cannot solve the problem of "cold start" [6], and it cannot provide accurate recommendation results when new users or new items are added.
- Collaborative filtering-based recommendation method relies on grouping items and users together through clustering, which has made it a widely used approach. Collaborative filtering offers two types of filtering options: nearest-neighbor and model-based. Nearest neighbor-based filtering is divided into two categories: user-based and item-based. User-based filtering connects you with like-minded individuals who share similar interests. Information about users is collected, their similarity is calculated using nearest neighbor search, and recommendations are generated based on the rating data and similarity matrix. Instead of comparing users, item-based collaborative filtering looks at the similarities between items. As the number of users and items grows, data sparsity increases, making nearest-neighbor-based collaborative filtering algorithms more complex. Collaborative filtering methods use behavioral data to identify features and match users with relevant items. The models include association-based, classification-based, clustering-based, and hidden factor models. (LFM) [7]. Among them, the singular value decomposition (SVD) [8], non-negative matrix (NMF) decomposition [9], and probability matrix factorization (PMF) [10] methods have achieved remarkable results in the Netflix Recommender System Competition. The hidden factor model (LFM) uses training suitable hidden factors based on the existing preference data to mine out the hidden features between users and items.

Although both of these methods have their strengths, they each have limitations when it comes to utilizing the available data. Hybrid recommender systems, on the other hand, combine multiple algorithms to enhance the effectiveness of recommendations. While these hybrid approaches can lead to more precise predictions, the challenge lies in achieving a proper balance of the results from each individual model. Recommendation models based on deep learning have gained widespread attention due to the rise of this technology. In comparison to traditional methods, neural networks can more precisely identify user and item characteristics and possess a strong resistance to noise.

- DSSM: In 2013, the Deep Semantic Model (DSSM) was introduced as the earliest method that incorporated textual information into deep recommendation models [11]. This model learns the similarities between users and items using neural networks and semantic representations. The DSSM uses a neural network to transform text data on users and items into a low-dimensional vector space and calculate their similarity. Since then, the proposal of the DSSM model has paved the way for the development of subsequent recommendation models for textual information, and it has become one of the milestones in the field of deep learning recommendation.
- DeepFM: The DeepFM model [12] is a type of recommendation model that is based on neural networks and Factorization Machine (FM) [13]. It was proposed by the authors of Huawei Noah's Ark Lab. Just like the traditional FM model, DeepFM also combines input features in a second-order cross-combination. However, DeepFM is different in that it also includes a neural network component

to capture the nonlinear interactions among the features. Compared to the traditional FM model, DeepFM has the advantage of capturing richer feature interaction information and better nonlinear modeling capability. Additionally, DeepFM is better at processing sparse features due to the utilization of low-dimensional hidden vector representation.

- NFM: The NFM model uses a feature crossover approach similar to the FM model [14]. However, unlike traditional factorization machines that model crossover features through vector inner products, the NFM uses a fully-connected neural network to learn the feature crossover representations. This allows the NFM to better capture the nonlinear relationships between features and adaptively learn the weights between them. Compared to other deep recommendation models, the NFM has lower computational complexity, faster training speed, and better recommendation performance.

This study proposes integrating textual information with the TwinBert model to improve recommender systems [15]. This model takes in textual data of users and items, which are fed separately into two Bert models. Non-textual data is also included in the model and converted into serial number encoding. The textual and non-textual data is then fused through a fully connected layer with a feed-forward neural network to create two dense user and item feature embedding matrices. These matrices are then pointwise multiplied and mapped to the interval [0,1] using the sigmoid activation function to obtain the predicted score. During training, the predicted scores are compared to the actual scores to improve the model's performance [16]. For validation, this study uses the recall metric to calculate the average movie embedding for each input user based on highly rated movies they have watched [17]. To provide users with accurate movie recommendations, one calculates the cosine similarity between their movie preferences and the database of movies. This method not only takes into account their previous movie ratings but also utilizes textual data to enhance the accuracy of our recommendations.

2. Data and Method

Amazon Customer Reviews Dataset (ACRD) is an open-source dataset provided by Amazon [18], which is divided into two main parts: one part is the metadata of the products, and the other part is the data of customer reviews. The Software project (ACRD-S) was chosen for the experiments and its details are shown in Table 1. According to the data provided in the dataset, there are 8,774,662 users in the Software along with 82,684 items and 2,997,348 reviews. The sparsity percentage is 99.853%, which implies that the dataset is highly sparse, and a significant number of products have not been reviewed by users. Since the Amazon review dataset contains a large number of missing, duplicate and erroneous values and may contain a large number of irrelevant or redundant features, it needs to be initially processed to ensure model training. The format of the processed portion of the data is shown in Table 2.

Table 1. ACRD - S Information

Categories	Number of Users	Number of Items	Number of Reeviews	Remoteness /%
Children’s	17567	3853	72432	99.93
Accounting	214	678	204073	99.91
Antivirus	39801	2837	32372	99.93
Business	216227	4615	1288357	99.99
Design	956	502	4811	99.89
Education	246166	46642	2218511	99.60
Lifestyle	11230	4562	33190	99.99
Music	41238	28267	3500139	99.99
Networking	7073	6082	70712	99.60
Operation System	6755	5817	201435	99.76
Photography	259583	3733	3219049	99.89
Programming	12714	22773	380794	99.96
Utilities	18426	18576	416516	99.95
Video	1020472	176009	1790394	99.82

Table 2. Format of Partially Processed ACRD-S

User ID	Item ID	Comment Text	Rate
A330QCZ2OYJ10C	B000AOJOUI	[101, 2113, 2210, 4773, 2640, 2565, 12476, 220...	0.2
A38NELQT98S4H8	0321719816	[101, 2224, 3959, 8545, 22208, 8646, 26632, 16...	0.75
A1I1KK110CQAVQ	B00005AC56	[101, 2052, 2507, 2053, 2732, 2071, 2448, 4007...	0.0
A3QJU4FEN8PQSZ	0321719816	[101, 9703, 2079, 7473, 2544, 4431, 6097, 2544...	0.75
ACJT8MUC0LRF0	0321719816	[101, 2215, 4553, 3443, 4037, 2593, 3768, 7023...	1.0
A2VVNXK8Y15RUT	B00068IBZW	[101, 2028, 2565, 7473, 6895, 21202, 5672, 236...	0.2

When it comes to data preprocessing, it's important to keep a close eye on the length of comment text. This involves filling the text with additional content if it's too short, and truncating the text if it's too long, to ensure consistent length of training data. Additionally, index columns need to be added for classification features to facilitate input to the neural network. The input data also needs to be converted to a fixed format for ease of use. To create a validation set for training, 10% of samples are randomly selected from the original dataset.

MovieLens is a widely used dataset that contains ratings and metadata about movies [19]. The GroupLens research group at the University of Minnesota created a dataset to study recommender systems and collaborative filtering algorithms. The dataset has multiple versions, with the MovieLens 100K dataset being the earliest and most popular version. Within this data, there exist 100,000 ratings for 1, 682 movies, given by a total of 943 users. The MovieLens 1M dataset has more users and movies, containing 1,000,000 rating records for 3,952 movies by 6,040 users. Table 3 provides a visual comparison of the two datasets. Although the 1M dataset requires more computational resources and time to process and load than the 100K dataset, it is more suitable for training and evaluating recommender systems. The details of the MovieLens 1M (ML-1M) dataset are shown in Table 4. The data format in the Movie Lens dataset may not be consistent with our analysis needs, therefore, each feature item needs to be indexed and scores need to be normalized, and the text needs to be converted into coded form. In addition, the irrelevant items Timestamp and Zip-code need to be discarded. The processed data is shown in Table 5. The dataset underwent division into three sets - training, testing, and validation - in an 8:2:1 ratio. The random_state parameter was used to ensure consistency of results across divisions.

Table 3. Description of MovieLens Dataset Information

Dataseet	Number of Users	Number of Items	Number of Reeviews	Remoteness /%
100K	943	1682	100000	93.70
1M	6040	3706	1000209	95.53

Table 4. User Scoring Information for the ML-1M Dataset

Rating Scale	ML-1M
1.0	56174
2.0	107557
3.0	261197
4.0	348971
5.0	226310
Total	1000209

Table 5. Format of Partially Processed ML-1M Dataset

UserID_idx	MovieID_idx	Rating	Gender_idx	Age_idx	Occupation_idx	Genres_idx	Title_encoding
320	2259	0.0	1	6	12	9	[101, 6144,...
403	3317	0.75	1	6	12	2	[101, 1046,…
2287	6	0.0	1	0	0	0	[101, 21876,...
2813	1813	0.75	1	5	3	3	[101, 26631,...
5674	2634	0.5	1	5	11	2	[101, 3714,…
5850	3538	1.0	0	6	4	2	[101, 2028,…

2.1. Method

Figure 1 outlines the general structure of the model, which is based on the DSSM double-tower model. The model is divided into four main components: input layer, embedding layer, feature transformation layer, and output layer. It models the user and item separately by utilizing text and other features to obtain feature matrices for each. These matrices are then input into a multilayer neural network to obtain dense vectors for the user and item, which are finally multiplied and mapped to a range of 0 to 1 to generate the final score. The following sections will provide a detailed illustration of the model. At the input stage, the features are categorized into four groups: user text information u_{text}, user other features u_n, item text information i_{text}, and item other features i_m. Here, n and m represent the total number of user and item other features, respectively. For instance, taking the MovieLens dataset as an example, user ID, gender, age, and occupation are classified as non-text features that belong to other features. Similarly, movie ID and type are also considered other features, while movie title belongs to the text features category. In the embedding layer, a fully connected operation is required after Index encoding or Bert embedding encoding. To represent features in the model, the index encoding assigns a unique integer to each distinct feature value [20]. For user and item features u_n and i_m, serial number encoding is utilized to transform them into embedding vectors. The formula for encoding feature u_1 is as follows:

$$f_{u_1} = \begin{cases} 0 & u_1 = v_0 \\ 1 & u_1 = v_1 \\ \vdots & \vdots \\ k-1 & u_1 = v_{k-1} \end{cases} \tag{1}$$

In the given feature u_1, there are several different fetches denoted by v_0, v_1, ..., v_{k-1}, where k represents the total number of different fetches. The sequential encoding process assigns a unique integer to each fetch, incrementing from 0 to $k-1$. The final output is the encoded feature vector f_{u_1}.

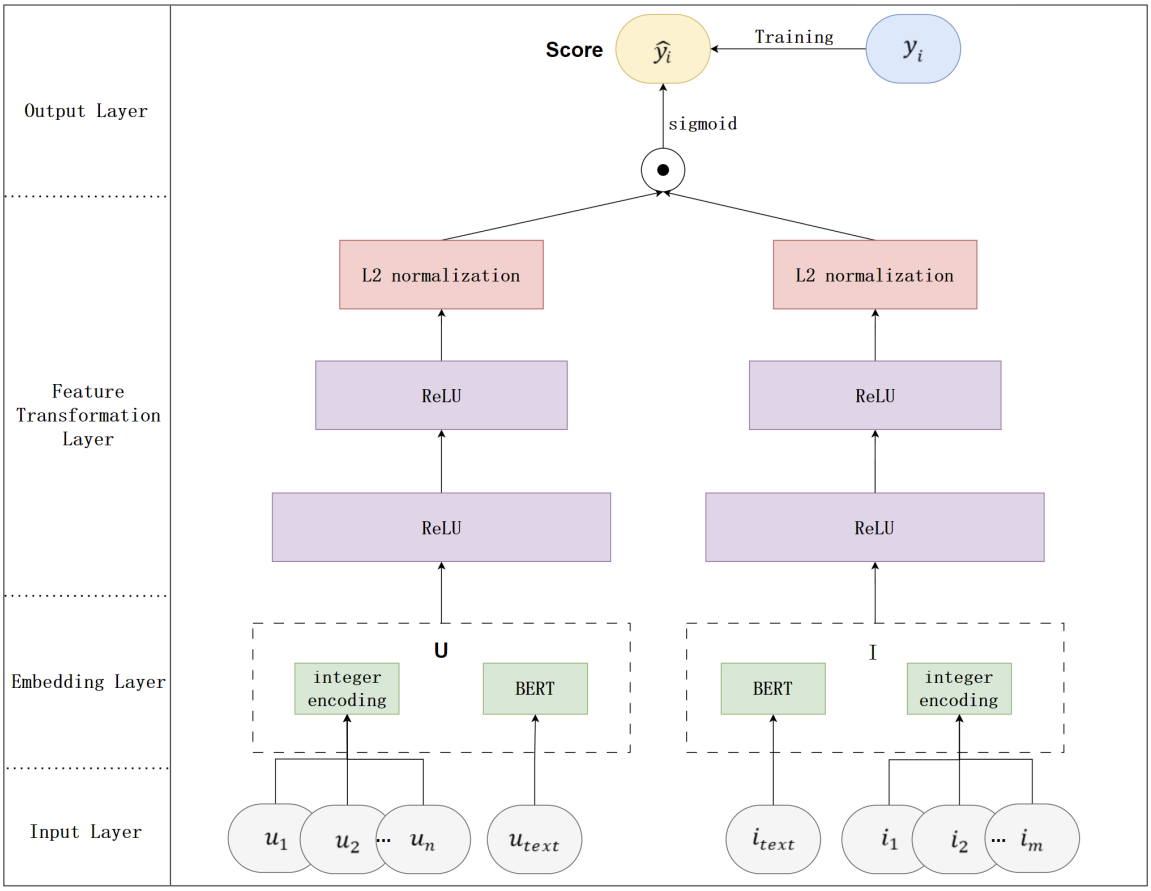


Figure 1. Overall framework diagram of the model (Photo/Picture credit: Original).

Bert embedding is a technique that utilizes the Bert model to encode text information in a dataset. The basic idea is to use the Transformer structure to encode input text and capture contextual information through a multi-layer self-attention mechanism [21]. This mechanism helps to understand the text better. Equation (2) describes the text embedding process of the Bert model using user text features as an example.

$$BERT(u_{text1}, u_{text2}, \dots, u_{textn}) = [E(u_{text1}), E(u_{text2}), \dots, E(u_{textn})] = E_{u_{text}} \quad (2)$$

The input text data is represented by $u_{text1}, u_{text2}, \dots, u_{textn}$, while $E(u_{text_i})$ represents the word vector corresponding to each word u_{text_i}. The $BERT(\cdot)$ model combines all the word vectors of the input text into a matrix, which is then outputted as $E_{u_{text}}$. This output matrix is used for subsequent tasks related to the input text. Similarly, the item text feature undergoes the same operation as in Equation (2) to obtain $E_{i_{text}}$. To create user and item feature vectors, one needs to combine various features. These include the text-like features $E_{u_{text}}$ and $E_{i_{text}}$, as well as non-text-like features f_{u_n} and f_{i_m}. Once one encodes these features, one concatenates them to form the user feature vector U and item feature vector I.

$$U = Concat(f_{u_1}, f_{u_2}, \dots, f_{u_n}, E_{u_{text}}) \quad (3)$$

$$I = Concat(f_{i_1}, f_{i_2}, \dots, f_{i_m}, E_{i_{text}}) \quad (4)$$

After the connections, the user and item sparse vectors are inputted into the feature transformation layer. The feature transformation layer is a component that converts sparse feature vectors into fixed-length dense feature vectors. This model has two fully connected layers with ReLU activation

functions used in each layer [22]. Additionally, the second fully connected layer uses L2 regularization to avoid overfitting and constrain the model [23].

$$F_U = Dense(Dense(U, 32, ReLU), 8, ReLU) \tag{5}$$

$$F_I = Dense(Dense(I, 32, ReLU), 8, ReLU) \tag{6}$$

Where $Dense(\cdot)$ represents the fully connected layer, U and I are low-dimensional representations of user and item features, and F_U and F_I are dense representations of user and item features. The output layer calculates the predicted score $\hat{y}_i$ by combining the user-dense features and item-dense features obtained from the feature transformation layer. Next, the predicted score is compared with the actual rating provided by the user for that particular item. In case of any discrepancies, the parameters in the previous layers are updated and iterated using backpropagation to improve the accuracy of the predictions.

$$\hat{y}_i = \sigma(F_U \cdot F_I) \tag{7}$$

Here, σ denotes the $sigmoid$ function. In addition, the Rating column is normalized to map each rating to a real value between 0 and 1, which facilitates subsequent recommendation model training.

$$y_i = (rating - min\_rating) \,/\, (max\_rating - min\_rating) \tag{8}$$

min_rating and max_rating are the minimum and maximum values in the Rating column, respectively, while $rating$ represents the raw rating value. The model uses back-propagation to adjust the weights and biases in order to minimize the mean square error.

$$loss = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{9}$$

Where n is the number of samples, y_i is the true value, and $\hat{y}_i$ is the predicted value of the model score. To evaluate and analyze a regression model for different datasets, this study has chosen the mean square error (MSE) as the loss function [24], and more generalized mean absolute error (MAE) [25], root mean square error (RMSE) [26], and Cosine similarity [27] between the embedding vectors of input users and the embedding vectors of the recommended items as the evaluation metrics. Below are the formulae for calculating mean square error, mean absolute error, root mean square error, and similarity score.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{10}$$

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{11}$$

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{12}$$

$$cos(\theta) = \frac{U_I \cdot I_P}{\|U_I\| \cdot \|I_P\|} \tag{13}$$

Where n represents the number of samples, y_i represents the true value, $\hat{y}_i$ represents the predicted value, U_I represents the embedding vector of the input user, I_P represents the embedding vector of the predicted item, $\|\cdot\|$ represents the modulus of the vector, and θ represents the angle

between U_I and I_P. Equation (13) computes the result as a value ranging between [-1,1], which indicates the similarity between the two vectors. When 'cos(θ)' is 1, it means that the two vectors are identical; when $cos(\theta)$ is -1, it means that the two vectors are completely opposite; and when $cos(\theta)$ is 0, it means that the two vectors are completely unrelated. The experimental model used a BERT preprocessing model to initialize the text information. The training process involved 40 iterations, with a batch size of 32 for each iteration. The user and item embeddings were both set to a dimension of 32. RMSprop was used as the optimizer with a learning rate of 0.001.

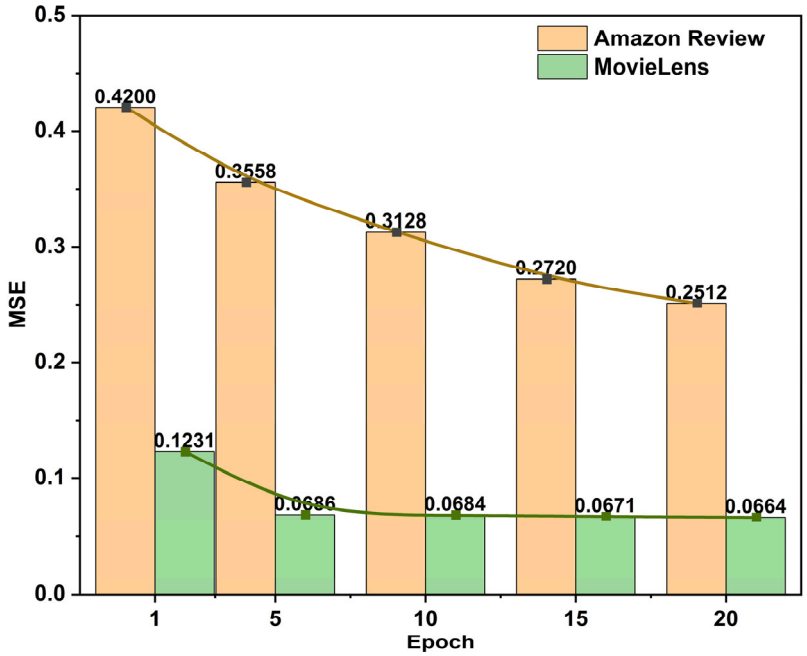


Figure 2. Comparison of Experimental Results for Loss Function MSE (Photo/Picture credit: Original).

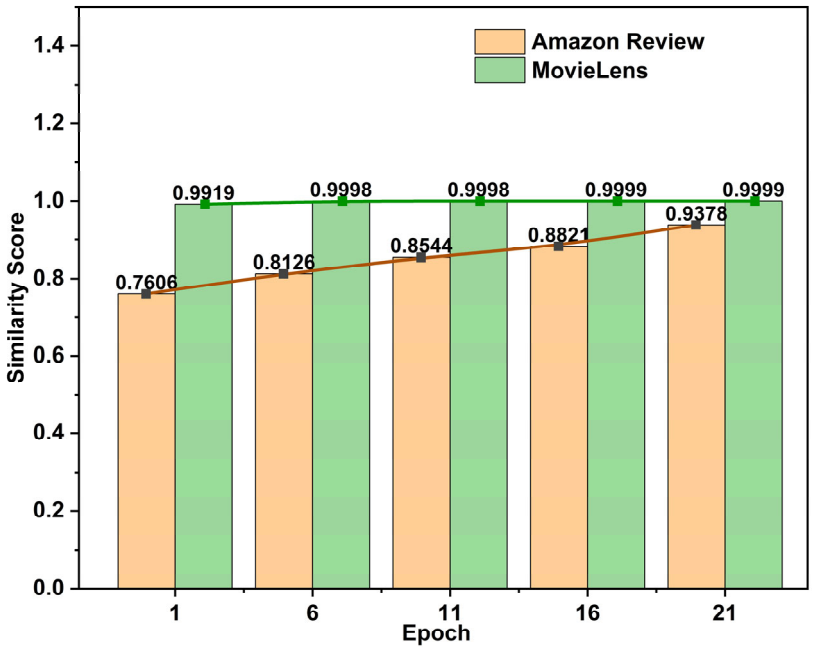


Figure 3. Comparison of Experimental Results on Similarity Scores (Photo/Picture credit: Original).

3. Results and Discussion

3.1. Comparison of Datasets

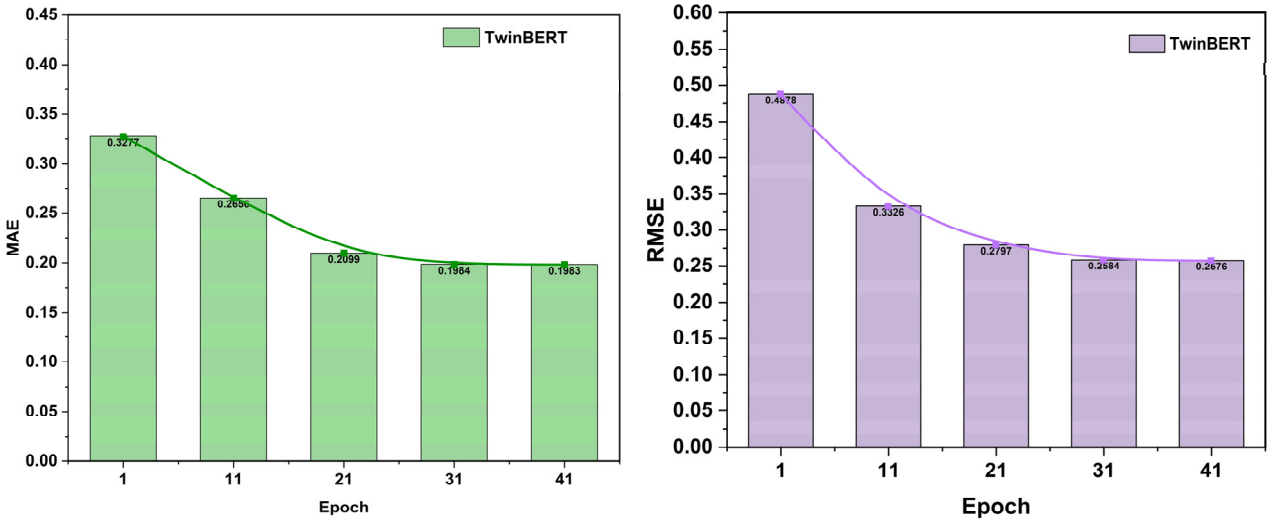
In the experimental analysis section, the model's performance is evaluated on two different datasets - the Amazon Customer Reviews Dataset of Software items (ACRD-S) and the MovieLens dataset (ML-1M). Seen from Fig. 2, Fig. 3, Table 6 and Table 7, the model is trained on each of the two datasets and its performance is compared by evaluating the Root Mean Square Error (MSE) of the loss function and the similarity scores (Cosine) of the input users and the recommended items during training. Upon comparing the evaluation metrics and experimental findings of the ACRD-S dataset and the ML-1M dataset, it is evident that the current model outperforms on the ML-1M dataset. This is primarily due to the fact that the ML-1M dataset has less sparse data and contains more information about the user's features. In contrast, the model performs relatively poorly on the ACRD-S dataset.

Table 6. Comparison of Experimental Results for Loss Function MSE

Dataset	Epoch 1	Epoch 5	Epoch 10	Epoch 15	Epoch 20
ACRD-S	0.42	0.3558	0.3128	0.272	0.2512
ML-1M	0.1231	0.0686	0.0684	0.0671	0.0664

Table 7. Comparison of Experimental Results on Similarity Scores

Dataset	Epoch 1	Epoch 5	Epoch 10	Epoch 15	Epoch 20
ACRD-S	0.760626	0.8126316	0.8544452	0.8821013	0.9377595
ML-1M	0.9918663	0.99976516	0.9997652	0.99986516	0.9999102



(A) (B)

Figure 4. Performance of the TwinBERT Model in Iterative Rounds. (a) MAE of TwinBERT, (b) RMSE of TwinBERT (Photo/Picture credit: Original).

3.2. Comparison of Models

The effectiveness of the models on the ML-1M dataset was measured using two evaluation metrics, MAE and RMSE. The baseline models consisted of Item-based model [28], SVD model [29], FunkSVD model [28], StaTNA model [30], MLP-Genetic algorithm [31], Generalized feed-forward network [31], ILIG model [32], FM model [33] with AFM model [33]. The model used in this paper is TwinBERT. The comparison results with the baseline model can be seen in Table 8. The TwinBERT model shows improvement over most baseline models for recommender systems, but is still inferior to the leading AFM model. The performance of this model in the training process and recommendation results is shown next. Figure 4 demonstrates a gradual decrease of both MAE and RMSE for the TwinBERT model with increasing iterations, stabilizing at 0.1984 and 0.2576,

respectively. Figure 5 illustrates that the highest Cosine score of 0.9998 is achieved between 21-31 epochs, indicating that the recommended movie is highly likely to be enjoyed by the user.

Table 8. Comparison of TwinBERT Model and Baseline Model Results

ML-1M Dataset		
Method	MAE	RMSE
Item-based	0.6893	0.8850
SVD	0.6858	0.8738
FunkSVD	0.6729	0.8637
StaTNA	0.5750	0.7520
MLP-Genetic algorithm (3 hidden layers, gradient)	/	0.4984
Generalized feed-forward network (3 hidden layers, momentum)	/	0.4632
ILIG	0.4261	/
FM	/	0.2681
TwinBERT (paper model)	0.2099	0.2673
AFM	/	0.1686

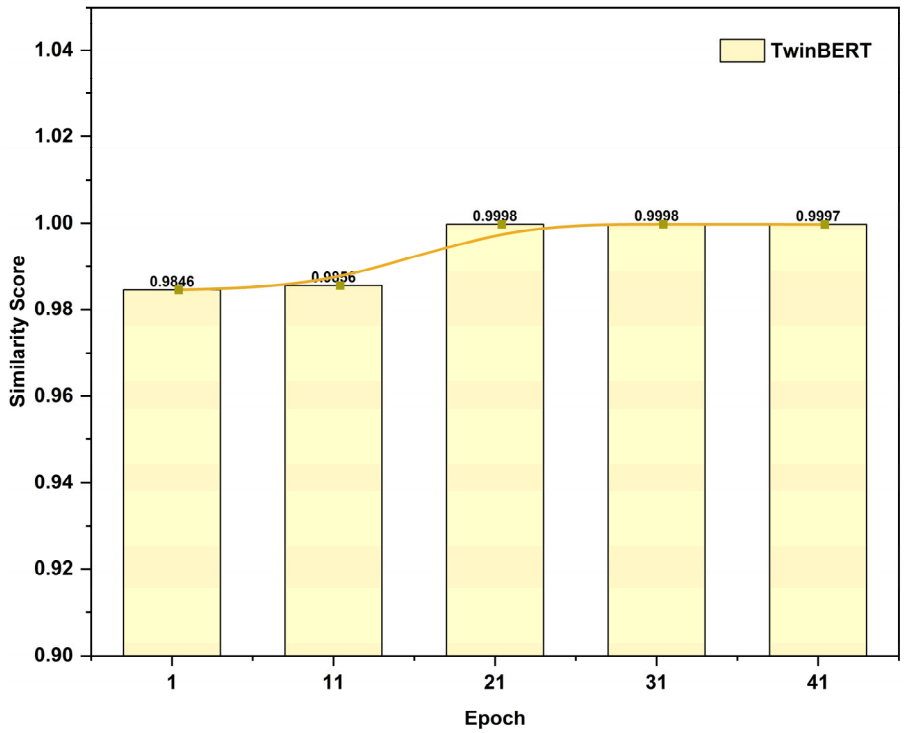


Figure 5. Results of Cosine Function Evaluation (Photo/Picture credit: Original).

4. Limitations and Prospects

Although the TwinBERT model improved the recommender system results compared to the baseline model, there are still issues to be resolved in the future. The TwinBERT model alone is not enough to solve the cold start problem. To address this, one can explore other data sources like the user's social network information and browsing history to initialize new users. Additionally, deep learning generative models such as Generative Adversarial Networks (GANs) [34] or Variational Autoencoder (VAE) can be utilized to generate embedding vectors for new users and items to overcome the cold start problem [35]. The TwinBERT model is still facing issues while dealing with sparse data. To address this problem in the future, one can consider augmenting user and item feature information. Additionally, onecan explore some special data processing methods such as clustering-based techniques [36] or recommendation algorithm-based approaches to fill in the missing data.

Moreover, one can attempt deep learning-based matrix decomposition methods like the Neural Collaborative Filtering (NCF) model to better model and predict sparse data.

5. Conclusion

In recommender systems, processing text data can be a challenge. To address this, the king model uses the TwinBERT model, which has better semantic representation ability and can extract more feature information. It also has better migration learning ability and supports multiple languages. However, there are still issues with the TwinBERT model when it comes to cold start and data sparsity. In the future, the model can be improved by introducing more data sources, feature information, and better methods.

References

[1] H. Li, J. Yao and C. Chen, arXiv preprint arXiv:1205. 6700 (2012).

[2] Q. Feng, and R. Cai, Journal of World Architecture, 5 (6), 52 - 61 (2021).

[3] G. Adomavicius and A. Tuzhilin, IEEE transactions on knowledge data engineering, 17 (6), 734 - 749 (2005).

[4] A. Xu and M. Raginsky, Advances in neural information processing systems, 30 (2017).

[5] C. F. G. D. Santos and J. P. Papa, ACM Computing Surveys (CSUR), 54 (10s), 1 - 25 (2022).

[6] J. Gao, G. Tian, A. Sorniotti, A. E. Karci and R. di Palo, Applied Thermal Engineering, 147, 177 - 187 (2019).

[7] Y. Koren, ACM Transactions on Knowledge Discovery from Data, 4 (1), 1 - 24 (2010).

[8] 8. Ben-Shimon, D., Rokach, L., & Shapira, B. (2016). An ensemble method for top-N recommendations from the SVD. Expert Systems with Applications, 64, 84 - 92.

[9] T. Liu, and D. Tao, IEEE Transactions on Neural Networks Learning Systems, 27 (9), 1851 - 1863 (2015).

[10] X. Ren, M. Song, E. Haihong, and J. Song, Neurocomputing, 241, 38 - 55 (2017).

[11] P. Huang, X. He, J. Gao, L. Deng, A. Acero and L. Heck, Proceedings of the 22nd ACM international conference on Information & Knowledge Management (2013).

[12] H. Guo, R. Tang, Y. Ye, et al., arXiv preprint arXiv. 04247 (2017).

[13] S. Rendle, 2010 IEEE International conference on data mining (2010).

[14] E. Martínez-Morillo, C. Childs, B. P. García, et al., Neurofilament medium polypeptide (NFM) protein concentration is increased in CSF and serum samples from patients with brain injury. Clinical Chemistry Laboratory Medicine, 53 (10), 1575 - 1584 (2015).

[15] W. Lu, J. Jiao and R. Zhang, Proceedings of the 29th ACM International Conference on Information & Knowledge Management (2020).

[16] B. Batou, Steel Composite Structures, An International Journal, 33 (5), 699 - 716 (2019).

[17] B. P. Staresina and M. Wimber, Trends in cognitive sciences, 23 (12), 1071 - 1085 (2019).

[18] R. He and J. McAuley, Proceedings of the 25th international conference on world wide web (2016).

[19] F. M. Harper and J. A. Konstan, ACM transactions on interactive intelligent systems, 5 (4), 1 - 19 (2015).

[20] H. P. Luhn, IBM Journal of research and development, 1 (4), 309 - 317 (1957).

[21] S. Abnar and W. Zuidema, arXiv preprint arXiv. 00928 (2020).

[22] V. Nair and G. E. Hinton, Proceedings of the 27th international conference on machine learning (ICML-10) (2010).

[23] C. Cortes, M. Mohri, and A. Rostamizadeh, arXiv preprint arXiv:1205.2653. (2012).

[24] D. M. Allen, Technometrics, 1 3(3), 469 - 475 (1971).

[25] G. Li, W. Chang, and H. Yang, IEEE Access, 8, 141432 - 141445 (2020).

[26] T. Chai, and R. R. Draxler, Geoscientific model development, 7 (3), 1247 - 1250 (2014).

[27] T. Thongtan and T. Phienthrakul, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop (2019).

[28] P. Z. Salam and S. Najafi, DIVA, 17 (2016).

[29] NicolasHug. Surprise-recommender-systems (2017).

[30] W. Chen, Z. Huang, J. C. N. Liang and Z. Xu, Open Review, 11 (2023).

[31] B. Şeref, G. E. Bostanci and M. S. Güzel, Turkish Journal of Electrical Engineering and Computer Sciences, 29 (1), 62 - 77 (2021).

[32] F. Dai, X. Gu, Z. Wang, et al., Paper presented at the Proceedings of the 2021 International Conference on Multimedia Retrieval (2021).

[33] J. Xiao, H. Ye, X. He, H. Zhang, F. Wu and T. S. Chua, Attentional factorization machines: Learning the weight of feature interactions via attention networks. arXiv preprint arXiv: 1708. 04617 (2017).

[34] A. Aggarwal, M. Mittal and G. Battineni, International Journal of Information Management Data Insights, 1 (1), 100004 (2021).

[35] L. Girin, S. Leglaive, X. Bie, J. Diard, T. Hueber and X. Alameda-Pineda, arXiv preprint arXiv: 12595 (2020).

[36] X. Wu, Y. Lao L. Jiang, et al., arXiv preprint arXiv: 2210.05666. (2022).