Published online: 28 June 2024 DOI: 10.54254/2977-3903/8/2024078

Design of a news recommendation model based on collaborative filtering and LSTM

Yiyang Huang

SILC Business School, Shanghai University, 99 Shangda Road, Baoshan District, Shanghai, China

2097749210@qq.com

Abstract. This paper aims to design a news recommendation model based on user preferences to address the issues in recommendation systems under large datasets. Initially, four datasets—click_history, news, news_embedding, and user_predict—were integrated into a single table, followed by data cleaning and feature engineering. Due to the large volume of data, this paper proposes necessary data filtering for the training and testing sets, utilizing temporal data to construct user feature vectors and news feature vectors. One challenge is how to effectively integrate user preferences and news features into the model to avoid overfitting or underfitting. In the model design and building phase, different methods were attempted to merge the information of users and news. Ultimately, the user preference features were processed using a fully connected layer, and the news embedding vectors were handled using an LSTM model. These two data parts were then combined into another fully connected layer, using ReLU as the activation function and CELoss as the loss function. Subsequently, the model's hyperparameters were adjusted and evaluated, achieving favorable model performance. The prediction accuracy for recommending news to users in user_predict was calculated as an evaluation criterion. Finally, this paper proposes directions for generalization and optimization in three aspects: data processing, model design, and experimental design. This includes data processing methods, potential improvements or mechanisms that could be incorporated into the model, and hyperparameter tuning. The paper primarily proposes data filtering to solve the problem of excessive data scale, which may aid in addressing recommendation system issues under large-scale datasets.

Keywords: machine Learning, collaborative filtering, text classification, LSTM

1. Introduction

In this project, the task is to design a news recommendation model based on user preferences using four datasets: click_history, news, news_embedding, and user_predict.

The click_history table contains information on the news each user has clicked on, totaling 1,112,623 entries, including the user's status at the time of clicking. The news table features characteristics of the news items, such as category, release time, and word count, totaling 364,047 entries.

The news_embedding table contains embedding vectors processed from the first 250 words of each news item, also totaling 364,047 entries.

The user_predict table includes identifiers for all users who need news recommendations, totaling 466 entries.

The ultimate goal is to use click_history to train a machine learning model, learning each user's reading preferences, and to recommend 50 news articles to these users. A recommendation is considered successful if at least one of the 50 news items matches the user's preferences, with the aim to maximize the number of correctly recommended users.

2. Research challenges and research approach

2.1. Challenges

2.1.1. Dataset partitioning

The primary challenge in this project is the large volume of data, which current equipment capabilities cannot fully process. This makes the partitioning of training and testing datasets a significant challenge. In partitioning the datasets, it's necessary to manage their dimensions to ensure they can be processed and analyzed by the current equipment. Moreover, focus should not solely be on reducing dataset dimensions; key information might be lost during the selection process in the training dataset, potentially affecting the final model outcomes. Direct data reduction in the test dataset could lead to a confusion in evaluating the success of the recommendations due to missing data.

2.1.2. Model design and parameter tuning

A major challenge is determining when to integrate user preferences and news features into the model, given that inputs from two different dimensions, users and news, enter the model. Incorrect data integration could prevent the model from effectively mining user preference information and news content features, potentially leading the recommendation results to deviate significantly from actual user interests. This may also cause the model to overfit or underfit, reducing the accuracy of the recommendation system.

2.2. Research approach

Initially, data preprocessing is required to integrate the four datasets, with click_history as the primary base for analyzing user preferences and recommending news [1]. This involves data cleaning, such as removing missing and outlier values, and converting timestamp data into usable time data. Subsequently, feature engineering is necessary; at this stage, training and test datasets are partitioned, with essential data filtering performed on the training dataset. The test dataset is also analyzed to see if its volume can be reduced. Factors potentially influencing user behavior are extracted from time data, along with user's historical click records, preference information, and news category and content, to construct user feature vectors and news feature vectors. Then, model design and construction take place. Since inputs from users and news enter the model in different dimensions, attempts can be made to merge them before entering the model or during model training. These methods can be compared for accuracy to select the better approach. Loss function design is also part of this phase. Following this, the model is trained and evaluated on the training dataset, adjusting hyperparameters such as learning rate and number of training iterations to achieve optimal model performance. News is then recommended to users in user_predict, and the number of correct predictions is calculated as an evaluation criterion. The timing of integrating user and news information needs to be decided at this step. Finally, considerations for model scalability and optimization are discussed, analyzing potential improvements or mechanisms that could be incorporated into the model.

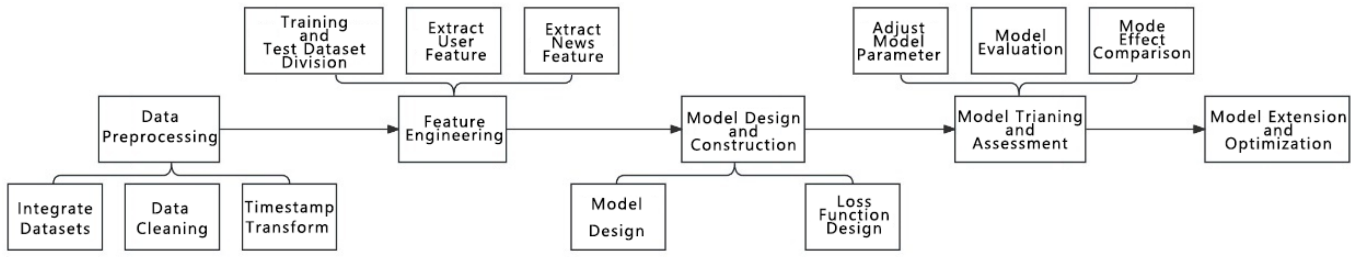

Figure 1. The research approach through a technical roadmap

3. Data Processing

3.1. Training and test dataset division

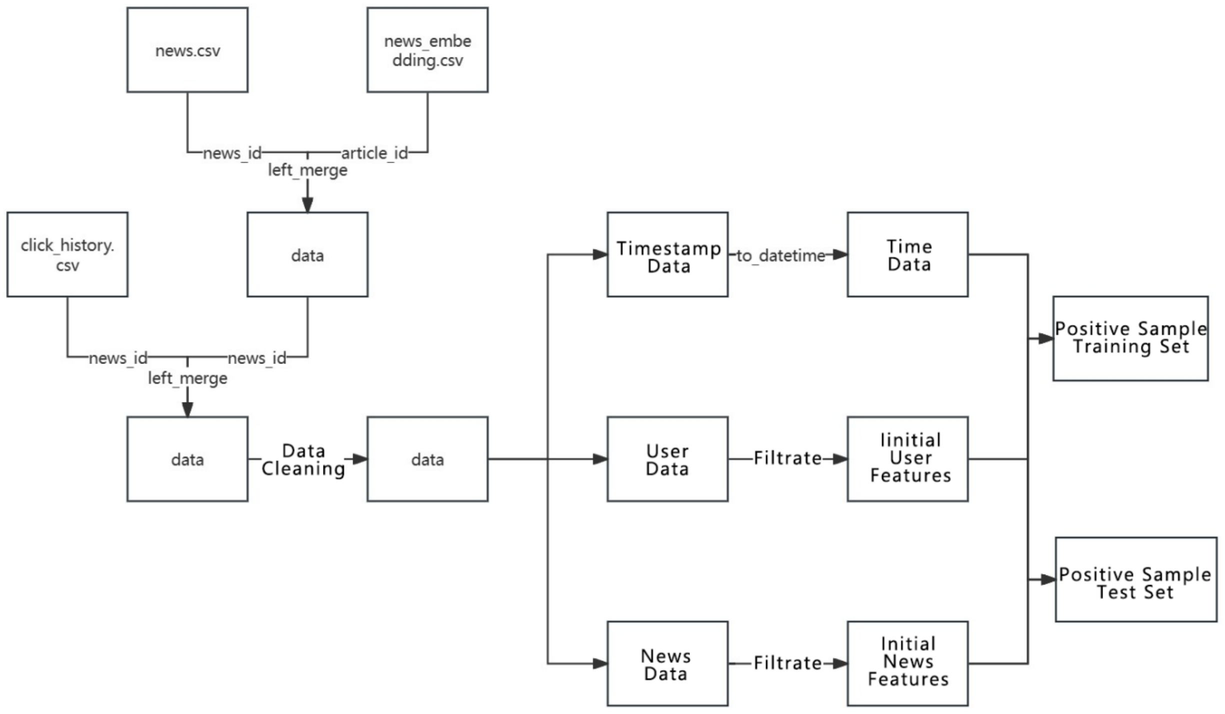

Figure 2. Training and test dataset division

Before dividing the training and test datasets, the given datasets need to be integrated into an initial dataset named 'data'. This dataset is then cleaned by removing rows with missing values and rows where 'create_time' is less than 'click_time', which are considered anomalies in the time data. Timestamp data is then converted into regular date-time data. Initial user features such as 'user_id', 'device', 'client_os', 'country', 'district', 'source', and 'context' are selected from the user data. Initial news features such as 'news_id', 'category', 'create_time', and 'word_num' are selected from the news data. It's important to note that the features selected at this stage might be lost in subsequent processes but will be replenished later, so this feature selection is only initial. Finally, the first 80% of the data is used as the initial positive sample training set, and the last 20% as the initial positive sample test set for use in later processes. The first five rows of the initial positive sample training set are shown in Table 1, and those of the test set in Table 2.

Table 1. Example of positive sample training set

User Id	News Id	Cateory	Create_time	Word_num	Context	Device	Client_os	Country	District	Source
199999	160417	281	2017/10/2 11:01	173	4	1	17	1	13	1
199999	5408	4	2017/10/3 1:30	118	4	1	17	1	13	1
199999	50823	99	2017/10/3 6:53	213	4	1	17	1	13	1
199998	157770	281	2017/10/2 22:38	201	4	1	17	1	25	5
199998	96613	209	2017/10/2 10:00	185	4	1	17	1	25	5

Table 2. Example of positive sample test set

User Id	News Id	Cateory	Create_time	Word_num	Context	Device	Client_os	Country	District	Source
196780	58580	118	2017/10/13 10:57	138	4	1	17	1	21	2
196780	271262	399	2017/10/13 10:09	182	4	1	17	1	21	2
196780	313920	431	2017/10/13 11:09	194	4	1	17	1	21	2
196780	236566	375	2017/10/13 10:56	192	4	1	17	1	21	2
196780	157798	281	2017/10/13 14:14	216	4	1	17	1	21	2

3.2. Training set processing

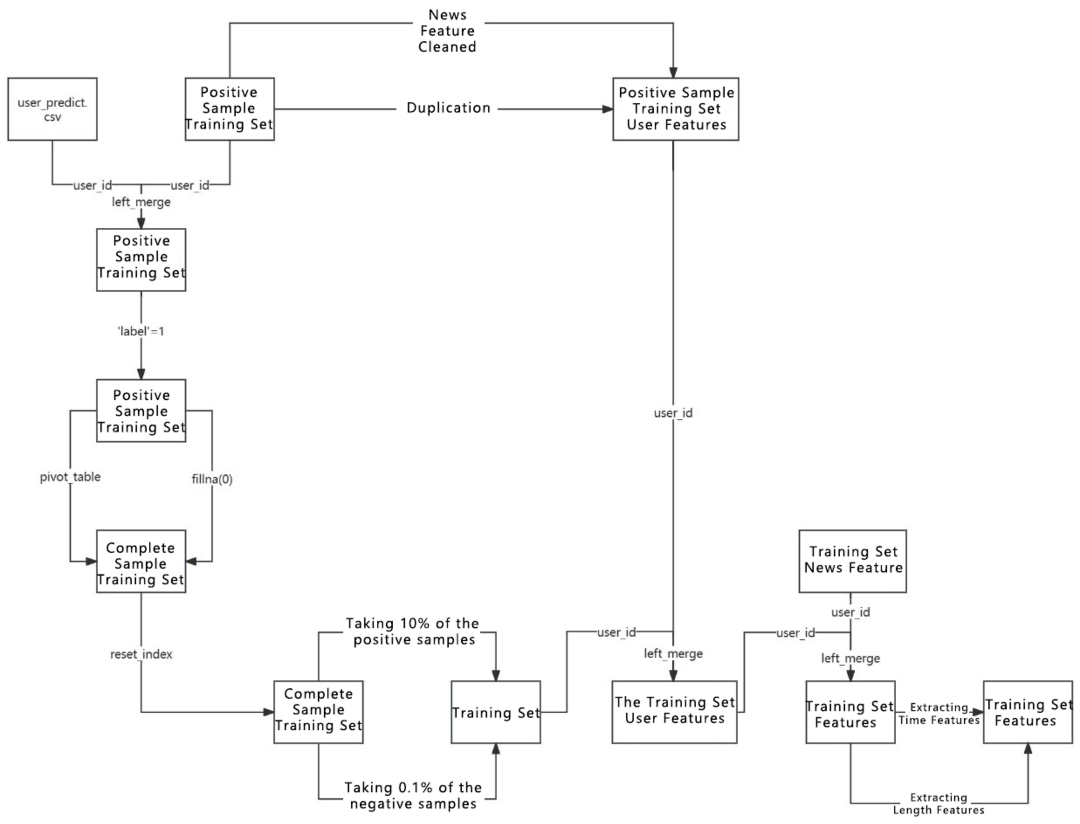
News Feature Cleaned
user_predict.csv
Positive Sample Training Set
Duplication
Positive Sample Training Set User Features
user_id
user_id
left_merge
Positive Sample Training Set
'label'=1
Positive Sample Training Set
pivot_table
fillna(0)
Complete Sample Training Set
user_id
Training Set News Feature
reset_index
Taking 10% of the positive samples
user_id
user_id
left_merge
user_id
left_merge
Complete Sample Training Set
Training Set
The Training Set User Features
Training Set Features
Extracting Time Features
Training Set Features
Taking 0.1% of the negative samples
Extracting Length Features

Figure 3. Training set processing

After obtaining the initial positive sample training set, as shown in Figure 3, it is processed to include negative samples and reduce the overall data volume. The initial positive sample training set is integrated into the user_predict dataset to identify needed entries and is labeled with a 1 to denote positive samples. A pivot_table method is then used to transform the 'user_id', 'news_id', and 'label' columns into a two-dimensional matrix, filling missing values with 0 to denote negative samples, thus creating a complete sample training set. The reset_index method is used to reshape the two-dimensional table back into its original format. The training set is then further reduced, taking 10% of the positive samples and 0.1% of the negative samples as the new training set. The initial positive sample training set, with news features removed and deduplicated, is integrated into the newest training set as the training set user features, while the training set news features are integrated into this dataset. Subsequent steps include extracting time features and length features, classifying time into months, days of the week, and different time segments of the day based on hours, and categorizing news length. The first five rows of the final training set are shown in Table 3; however, as there are 264 columns, it is impractical to display them all, so only the first 11 columns are shown here.

Table 3. Training set example

User Id	News Id	Cateory	Create_time	Word_num	Context	Device	Client_os	Source	Cateory	Create_month
32456	20691	1	4	1	17	1	25	1	9	10
32456	166123	0	4	1	17	1	25	2	289	10
32456	20691	1	4	1	17	1	25	2	9	10
32456	336245	0	4	1	17	1	25	1	437	10
32456	166123	0	4	1	17	1	25	1	289	10

Furthermore, this paper constructs a User-User matrix, an Item-Item matrix, and a User-Item matrix based on the similarity of features between users and news. Recommendations are made directly using the User-Item matrix, which has the advantage of not requiring detailed information about the users or items, only the interaction data between them. Multiplying rows and columns respectively provides the similarity between users and between items, which is used to make recommendations.

3.3. Test set processing

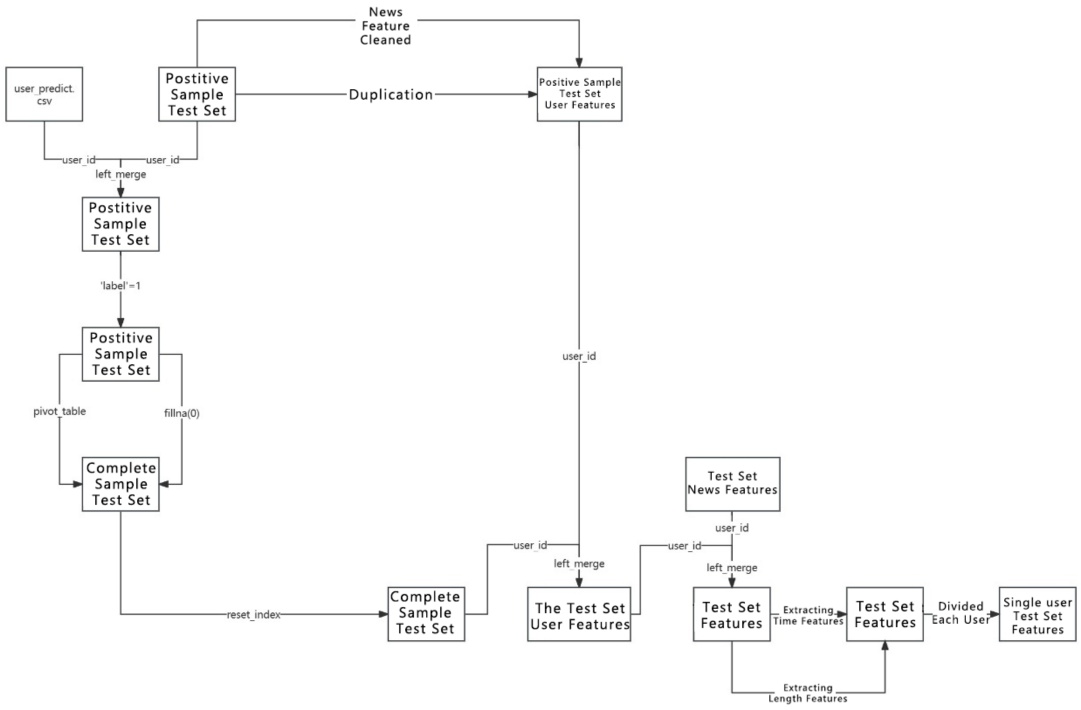

Figure 4. Test set processing

The approach to processing the test set is similar to that of the training set, but the test set cannot be reduced because doing so could lead to data loss that confuses the evaluation of recommendation success. Therefore, the test set is divided by 'user_id' into 466 separate datasets to manage the large size of the dataset. The other steps are consistent with those of the training set processing and are not reiterated here. The first five rows of the final test set are displayed in Table 4; as with the training set, there are 264 columns, so only the first 11 columns are shown.

Table 4. Test set example

User Id	News Id	Cateory	Create_time	Word_num	Context	Device	Client_os	Source	Cateory	Create_month
9720	119546	0	4	1	17	1	20	1	247	10
9720	157375	0	4	1	17	1	20	1	281	10
9720	159652	0	4	1	17	1	20	1	281	10
9720	96877	0	4	1	17	1	20	1	209	10
9720	335301	0	4	1	17	1	20	1	437	4

4. Model design

4.1. Principles of model design

The model used in this paper is based on fully connected operations and an LSTM model, consisting of five layers: input layer, user feature processing layer, news content processing layer, concatenation layer, and another fully connected layer.

In the input layer, the model receives two types of inputs: user features and the embedded representations of news content. User features represent personal information and historical news viewing data to reflect user preferences. The embedded representations of news content are derived from the first 250 words of the news content [2].

The user feature processing layer is a fully connected layer that transforms user features into a representation with hidden layer dimensions. Here, a ReLU activation function is used for nonlinear transformation. The formulas for the fully connected layer and the ReLU function are as follows, where W is the weight matrix, x is the input, b is the bias, y is the output, and f is the activation function. Each column of the weight matrix W represents the output features of the previous layer, each row represents the input features of the next layer, and each element represents the relationship between two features.

Fully Connected Layer Formula: $y = f(W \bullet x + b)$

ReLU Function Formula: $\mathrm{ReLU}(x) = \max(0, x)$

The news content processing layer is an LSTM layer that takes the embedded representations of news content as input. In this model, the output from the LSTM layer, along with user features, will serve as input for subsequent layers. The most important aspects of the LSTM are the three gates—input, output, and forget—and the two types of memory: long-term memory C and short-term memory h. The formulas are as follows [3]:

Input Gate Formula: $i_t = \mathrm{sigmoid}(W_{ii}x_t + b_{ii} + W_{hi}h_{(t-1)} + b_{hi})$

Output Gate Formula: $o_t = \mathrm{sigmoid}\left(W_{io}x_t + b_{io} + W_{ho}h_{(t-1)} + b_{ho}\right)$

Forget Gate Formula: $f_t = \mathrm{sigmoid}\left(W_{if}x_t + b_{if} + W_{hf}h_{(t-1)} + b_{hf}\right)$

$$g_t = \tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{(t-1)} + b_{hg}\right)$$

Long-term Memory Formula: $c_t = f_t \odot c_{(t-1)} + i_t \odot g_t$

Short-term Memory Formula: $h_t = o_t \odot \tanh(c_t)$

In the concatenation layer, the model concatenates the outputs of user features and news content on the feature dimension to form a new tensor.

Finally, another fully connected layer maps the concatenated tensor to the output dimensions and uses a ReLU activation function for nonlinear transformation to predict whether a user has seen a specific news item.

The comparative model used in this paper concatenates user features and news content before entering the model and processes them through fully connected and LSTM layers [4].

4.1.1. Feature extraction of text data

In the steps of data preprocessing and feature engineering, this paper has already performed certain feature extractions on text data. Now, it is only necessary to separate text features from user features and convert them into Tensor data, allowing them to be input into the multi-input model designed in this paper. Here, the torch.tensor method is used to convert data into Tensor data. Considering data type and size impacts, the last 250 entries of the dataset, i.e., the embedding vectors, are used as text features. To prevent overfitting, 'user_id' and 'news_id' are removed from the remaining user feature data, leaving 11 columns as user features.

4.1.2. Model loss function design

The problem in this paper is a binary classification issue, and either BCELoss or CELoss could be used as the loss function. However, due to dimensional mismatch issues arising from feature concatenation in the concatenation layer, CELoss is used during training. The formulas for BCELoss and CELoss are as follows:

BCELoss: $\sigma(x) = \frac{1}{1+e^{-x}}$

$$L = -\sum_{i-1}^{N}[y_i \bullet \ln\left(\sigma(x_i)\right) + (1 - y_i) \bullet \ln(1 - \sigma(x_i))]$$

CELoss: $\sigma(x_i) = \frac{e^{x_i}}{\sum_{k=1}^{N} e^{x_k}}$

$$L = -\sum_{i-1}^{N} y_i \bullet \ln\left(\sigma(x_i)\right)$$

5. Experimental validation

5.1. Model parameter settings

This section details the parameter design and values used in the model within this paper, which includes user_feature_dim, embedding_dim, hidden_dim, and output_dim.

user_feature_dim is set to 11, representing the 11 columns used as user features.

embedding_dim is set to 250, representing the 250 columns used as text features.

Considering the large size of the dataset, hidden_dim is set to 64, aiming to capture the data's characteristics as effectively as possible within the limits of memory capacity.

Since this is a binary classification problem, output_dim needs to map outputs to 0 and 1 to output values for these two targets.

5.2. Evaluation metrics

According to the problem statement, a correctly predicted sample should correspond to a piece of news that a user has actually viewed in the test set. In this paper, 50 news items are recommended to users in the test set who need news recommendations. If one of these 50 news items is an article that the user actually viewed in the test set, it is considered a correct recommendation for that user. The evaluation of the model is based on the number of correctly recommended users—the higher, the better. Since the loss function indirectly represents the model's accuracy for 0-1 classification, the classification accuracy is defined as the ratio of correctly recommended users to the total number of users needing recommendations [5].

This paper does not consider the number of hits within the 50 recommended news items for the same user as an evaluation metric to reduce the complexity of model comparisons.

The methods used in this paper do not consider recommending news already viewed by the user in the training set and remove it from the test set to avoid label anomalies, which could lead to high prediction accuracy due to model overfitting but fail to recommend news that users actually viewed in the test set.

Moreover, repeating recommendations in the test set effectively uses the known data from the test set, similar to the treatment of the training set. Therefore, the method used in this paper only makes one recommendation per test set, so there are not multiple classification accuracies on the test set.

5.3. Comparative experiments

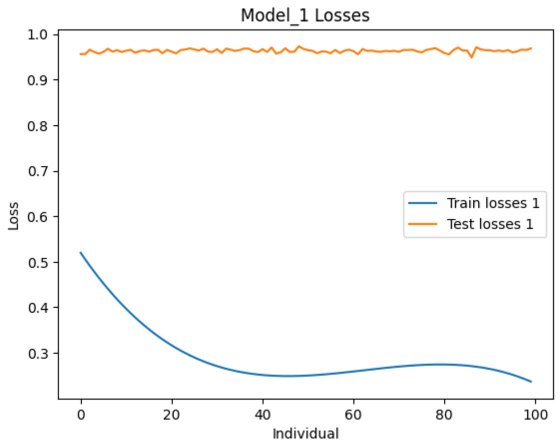

Figure 6. Loss during training and testing of the comparative model

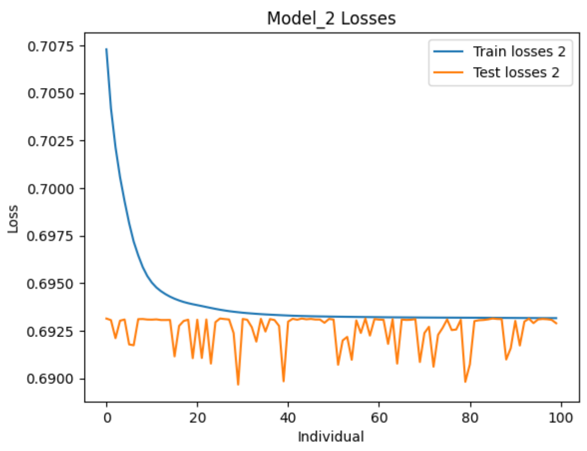

Figure 7. Loss during training and testing of the main model

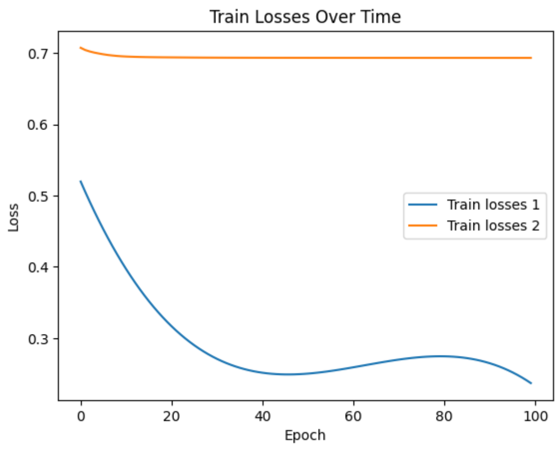

Figure 8. CELoss results of different models on the training set

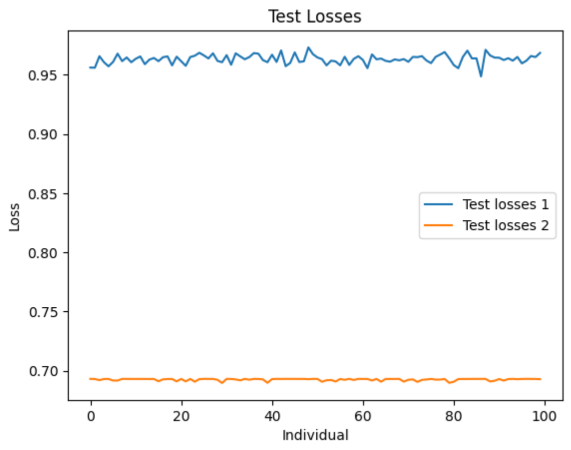

Figure 9. CELoss results of different models on the test set

In the above four figures, the comparative model is model_1, and the main model is model_2. According to the results, among 466 users, the comparative model's 50 recommendations hit 228 users, while the main model's 50 recommendations hit 404 users.

5.4. Analysis of experimental results

From the loss comparison and recommendation results in section 4.3, it is evident that although the main model's CELoss on the training set is significantly higher than that of the comparative model and the reduction in CELoss over 100 training iterations is smaller for the main model, the CELoss of the main model on the test set is lower than that of the comparative model. Additionally, the number of hits from the 50 recommendations is also higher for the main model, and the CELoss on the test set is lower than on the training set for the main model, suggesting effective model training. However, the chosen loss function might not be the best fit for the model, as indicated by the high loss values, suggesting that other loss functions could be explored. The comparative model's CELoss on the test set is higher than on the training set, and a rebound in CELoss on the training set suggests overfitting. Therefore, reducing the number of training iterations during experimental processes could prevent overfitting, making the model comparison more meaningful.

6. Future directions for improvement

The data processing methods, model design, and experimental design used in this paper all have areas that can be improved. The following will outline possible improvements for these aspects.

Firstly, in terms of data processing methods, since the datasets involved in this problem are large and current memory capacity is insufficient to learn from all user and training data, the training set was reduced by randomly selecting a portion of positive and negative samples. However, the selection of positive and negative samples could be based on certain user features to ensure that the model learns more features and user preferences during training. Ideally, with sufficient hardware conditions, the training set should not be reduced, and the test set should not be divided by users.

Secondly, in model design, introducing an attention mechanism could be beneficial to focus on features that are prominent and more meaningful for model training [6]. However, due to limited memory, the main model designed in this paper does not incorporate an attention mechanism. Nevertheless, considering other activation functions or transferring current user preference features like 'category', 'create_month', 'create_weekday', 'createtimeperiod', and 'word_num_label' to news features or as a third input into the model might be considered.

When choosing model evaluation standards, given that this problem is a binary classification issue, more effective evaluation metrics should be selected. Concepts like true positive rate and false positive rate could be introduced. Most importantly, to ensure that users' interests in news are not overlooked during recommendations, future improvements could consider using recall as an evaluation criterion, or introduce a Precision-Recall (P-R) curve.

Lastly, in the experimental design, multiple different loss functions could be used to more comprehensively compare model training effects, and appropriate hyperparameters should be selected during the experimental process to prevent overfitting and underfitting.

7. Conclusions

The main challenges encountered in this problem include learning about machine learning, processing data, and selecting and tuning models.

Before starting this research, I learned about various aspects of machine learning such as neural networks, collaborative filtering algorithms, GCN optimization for recommendation systems, and attention mechanisms [7]. During the study of collaborative filtering, I realized that user preferences might change over time, although collaborative filtering algorithms generally assume that preferences are stable [8]. Thus, methods need to be adopted to handle shifts in user preferences, such as using time-weighted techniques to consider recent interaction data. Additionally, collaborative filtering algorithms require selecting matching evaluation metrics, such as recall [9].

During data processing, due to limited memory on a laptop device, errors related to insufficient memory can occur during data integration. In such cases, the dataset needs to be filtered, and the method used in this paper was random selection, but improvements could consider selecting only a few users for news recommendations.

Regarding model selection, I initially planned to use a collaborative filtering algorithm directly [10]. However, considering that users viewing news might involve temporality, this paper combined LSTM with the collaborative filtering basis to address this issue [11].

References

[1] Tian, X., Ding, Q., Liao, Z. H., et al. (2021). A review of news recommendation algorithms based on deep learning. *Journal of Computer Science and Exploration, 15*(06), 971-998.

[2] Xue, C. X., & Zhang, Y. F. (2013). A review of Chinese text classification research in the news domain. *Library and Information Service, 57*(14), 134-139.

[3] Chai, Z. H. (2022). Bi-LSTM commodity recommendation system based on word embedding [Doctoral dissertation, Hebei University of Science and Technology]. https://doi.org/10.27107/d.cnki.ghbku.2021.000685
[4] S. N. K., & G. D. T. (2022). Dynamic light weight recommendation system for social networking analysis using a hybrid LSTM-SVM classifier algorithm. *Optical Memory and Neural Networks, 31*(1).
[5] Wu, G. D. (2021). Research on personalized item recommendation based on deep learning [Doctoral dissertation, Donghua University]. https://doi.org/10.27012/d.cnki.gdhuu.2020.000335
[6] Liu, G. (2023). Research on news recommendation methods based on knowledge graphs and personalized attention mechanisms [Doctoral dissertation, Hubei University]. https://doi.org/10.27130/d.cnki.ghubu.2023.000534
[7] Ma, H. W., Zhang, G. W., & Li, P. (2009). A review of collaborative filtering recommendation algorithms. *Journal of Mini & Micro Computer Systems, 30*(07), 1282-1288.
[8] Zhao, W., Lin, N., Han, Y., et al. (2016). An improved collaborative filtering algorithm based on K-means clustering. *Journal of Anhui University (Natural Science Edition), 40*(02), 32-36.
[9] Reham, A., Halah, A., & Amaal, A. (2023). Context-aware news recommendation system: Incorporating contextual information and collaborative filtering techniques. *International Journal of Computational Intelligence Systems, 16*(1).
[10] Yang, W., Tang, R., & Lu, L. (2016). News recommendation methods based on the integration of content-based recommendation and collaborative filtering. *Journal of Computer Applications, 36*(02), 414-418.
[11] Ai, P. Q. (2017). Research on personalized news recommendation systems based on temporal behavior and tag relationships [Doctoral dissertation, Tianjin University of Technology].