processes

MDPI

Article

Deep-Sequence–Aware Candidate Generation for e-Learning System

Aziz Ilyosov [1], Alpamis Kutlimuratov [1] and Taeg-Keun Whangbo [2,*]

[1] Department of IT Convergence Engineering, Gachon University, Sujeong-Gu, Seongnam-Si 461-701, Gyeonggi-Do, Korea; ilyosovaziz@gachon.ac.kr (A.I.); alpamis92@gachon.ac.kr (A.K.)
[2] Department of Computer Science, Gachon University, Sujeong-Gu, Seongnam-Si 461-701, Gyeonggi-Do, Korea
* Correspondence: tkwhangbo@gachon.ac.kr

Abstract: Recently proposed recommendation systems based on embedding vector technology allow us to utilize a wide range of information such as user side and item side information to predict user preferences. Since there is a lack of ability to use the sequential information of user history, most recommendation system algorithms fail to predict the user's preferences more accurately. Therefore, in this study, we developed a novel recommendation system that takes advantage of sequence and heterogeneous information in the candidate-generation process. The principle underlying the proposed recommendation model is that the new sequence based embedding layer in the model catches the sequence pattern of user history. The proposed deep-learning model may improve the prediction accuracy using user data, item data, and sequential information of the user's profile. Experiments were conducted on datasets of the Korean e-learning platform, and the empirical results confirmed the capability of the proposed approach and its superiority over models that do not use the sequences of the heterogeneous information of users and items for the candidate-generation process.

Keywords: recommendation system; candidate generation; deep learning; sequence-aware embedding

Citation: Ilyosov, A.; Kutlimuratov, A.; Whangbo, T.-K. Deep-Sequence–Aware Candidate Generation for e-Learning System. *Processes* **2021**, *9*, 1454. https://doi.org/10.3390/pr9081454

Academic Editors: Konstantinos Demertzis, Lazaros Iliadis, Nikos Tziritas and Panayotis Kikiras

Received: 22 June 2021
Accepted: 16 August 2021
Published: 20 August 2021

1. Introduction

Currently, recommendation systems play a crucial role in accelerating searches and allowing users to access more relevant content. Therefore, web service providers have been utilizing recommendation systems that analyze and harness user–item interactions to improve customer satisfaction and personalized recommendations and increase the income interests of their services. Moreover, with the development of deep-learning and machine-learning technologies, recommendation systems have become an integral part of multi-billion business organizations such as Amazon and Alibaba [1]. During the early development stages of recommendation systems, most recommendation models were based on the similarity concept. For example, the correlation (cosine similarity (1) or Pearson correlation (2)) of each item or user was calculated using historical data or content information, and items with higher correlations shared the same history that was not observed in the other item.

$$CS = \frac{AB}{||A||||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\ \sqrt{\sum_{i=1}^{n} B_i^2}} \quad (1)$$

$$PC = \frac{\sum_{i=1}^{n} (A_i - \overline{A}) * (B_i - \overline{B})}{\sqrt{\sum_{i=1}^{n} (A_i - \overline{A})^2} * \sqrt{\sum_{i=1}^{n} (B_i - \overline{B})^2}} \quad (2)$$

After the similarity-based concept, more accurate and fast latent-factor models were proposed. Based on the rating patterns, these algorithms explain user feedback by characterizing items and users based on factors, which comprise the features of items or users. For example, latent factors can represent the genre of music and refer to the values that point to the amount of action, comedy, and drama for movies. For users, each factor explains

the level of interest the user has in the feature of the item that the user rated with a high score. Matrix factorization is a popular realization approach of latent-factor models [2]. It defines item and user characteristics by a vector of factors based on a rating table. When the correlation between a user and an item is high, the item is recommended to the user. Due to their scalability and high accuracy, these methods have recently become highly popular. In addition, the flexibility of matrix factorization helps utilize real-life data. One advantage of matrix factorization is that it can use the feedback provided by all users as a rating matrix [3]. For example, when explicit ratings do not exist, the algorithm can derive the preference of the user using implicit feedback, which is made by observing the user's behavior without their direct click history or browsing history. The above recommendation concepts are classified as collaborative filtering and content-based filtering approaches for building recommendation models because of the type of information obtained; in these systems, the diversity of information affects their implementation and structure [4]. Collaborative filtering algorithms use the user's ratings of items to predict unrated items when making recommendations and then automate these predictions by gathering user expectations from a niche audience. In contrast, content-based filtering approaches provide suggestions by evaluating the availability of user–item interaction data, which entails handling a large amount of explicit data [5–7].

Deep Learning-Based Recommendations

The recent success achieved by deep-learning algorithms (convolutional neural networks, CNNs, and recurrent neural networks, RNNs) and frameworks (TensorFlow and Keras) has created a new era for recommendation systems. Because of the innovative prospects of deep-learning models, numerous deep learning-based recommendation systems have been recently proposed. The most attractive characteristics of neural models are that neurons are end-to-end differentiable and provide suitable inductive biases catered to the data type. If a model with a particular shape can be used for a certain data structure, it can be used for different data with similar structures. For example, CNNs and RNNs have long exploited structures in computer vision and natural language processing. Similarly, sequential structures of click-based or session-based logs are compatible with RNN and CNN. In addition, deep neural networks can be joined in a differentiable model and trained end-to-end. This helps in dealing with hybrid recommendation systems. For example, when text data (reviews, tweets) and images (posts, predict image) utilized in a recommendation deep-learning structure are incomparable choices. Here, traditional recommendation system models often fail and cannot utilize joint (end-to-end) representation learning. For instance, to process reviews, costly preprocessing (such as keyword extraction) is required, whereas text-based information can be directly input into deep learning-based models [8].

Deep and extensive exploitation of features of neural networks mostly helps utilize the side information in recommendation systems. In visual Bayesian personalized ranking [9], the visual appearance of a product is compressed into a one-dimensional feature vector using the associated CNN network and is concatenated to produce a full feature vector of positive and negative items in the Bayesian personalized ranking system. This method not only improves the accuracy of the network but also addresses the cold-start problem by providing an initial visual embedding feature of the item. When a system has an excessive number of users and items, the prediction time of preference increases linearly for both deep-learning and matrix factorization models. To solve this problem, candidate-generation (top-N recommendation) algorithms have been proposed [10,11]. However, in most cases, these models cannot utilize sequential and heterogeneous data (content information) simultaneously.

We have proposed a more accurate deep learning-based recommendation model that can utilize heterogeneous information and sequence data simultaneously. Specifically, the main contribution of our study is that the proposed model utilizes the sequence data of the user history and the item's heterogeneous information to improve the prediction

accuracy of deep-recommendation systems. To apply the sequence algorithm in deep-recommendation systems, we modified the first layer of the model by concatenating a vector that contains data regarding the order of items. This is because the average preference vector and sequential preference vector models can be aware of more information on the user, which leads to a more accurate recommendation.

In addition, concatenating extra information of the user to the first layer can help us overcome the cold-start problem. When information regarding the user–item interaction is insufficient, instead of the history vector, the default embedding vector is input, together with additional consumer data. This user information improves the recommender by providing an overview of the newcomers.

The remainder of this paper is organized as follows. In Section 2, the cold-start issues of recommendation systems, candidate generation, and deep learning-based recommendation models that involve heterogeneous information and sequence data are reviewed. In Sections 3 and 4, we detail the proposed model and verify its accuracy through experiments and comparisons with other similar recommendation models. The findings and scope of prospective work are presented in Section 5, and the examined materials are referenced, with many of them being recent publications.

2. Related Work

Several recent studies have applied sequence data and heterogeneous information as supplementary features to address the open issues related to candidate generation and cold starts in recommendation systems [12–16]. Lam et al. [13] developed a hybrid model based on the examination of two probabilistic perspective models utilizing unadulterated collaborative filtering to combine user data. Yu and Riedl [16] proposed a model that generates personalized stories using the optimal sequence of events experienced by the user to overcome the cold-start problem. Regarding user history, Zhao et al. [17] developed a product recommender system that considers the time interval between purchased products. Moreover, several studies have integrated different sources of information simultaneously into the prediction process [1,18,19].

In terms of tasks, most top-N recommendation systems are very similar to candidate-generation systems. Similar to candidate generation, the top-N recommendation systems filter items from large item spaces based on the precision of prediction. For example, in [11], the similarities of items were calculated, and item-based baskets were recommended to users. Another pioneering top-N recommendation system is AutoRec [20], which is built on a pure autoencoder algorithm. The model takes all user ratings as input data for the input layer and then compresses these data in the encoder part.

Starting from the bottleneck in the middle, the decoder part of the model was built. In this part, the layers of the model are expanded, and the initial dimension (input layer) in the output layer is restored (Figure 1).

$$h(r;\theta) = f(W \cdot g(Vr + \mu) + b)) \tag{3}$$

Here, $h(r;\theta)$ is the reconstruction of input r while f and g are activation functions. μ and b are biases in each layer.

$$\min_{\theta} \sum_{i=1}^{n} \left|\left| r^i - h(r^i;\theta) \right|\right|^2 + \frac{\lambda}{2}(||W||_F^2 + ||V||_F^2) \tag{4}$$

In AutoRec recommendation, the model is trained by minimizing the sum of the mean square errors between the output and input ratings and via L2 regularization, which handles the overfitting problem. In Equation (4), W and V are the weights of the first and second layers, respectively. r^i is the rating history of each user calculated using the equation. λ is the regularization parameter, which is recommended to be chosen by the cross validation technique. Despite the high accuracy of prediction and the flexibility with user history, autoencoder-based recommendation systems face some problems while handling user content data.

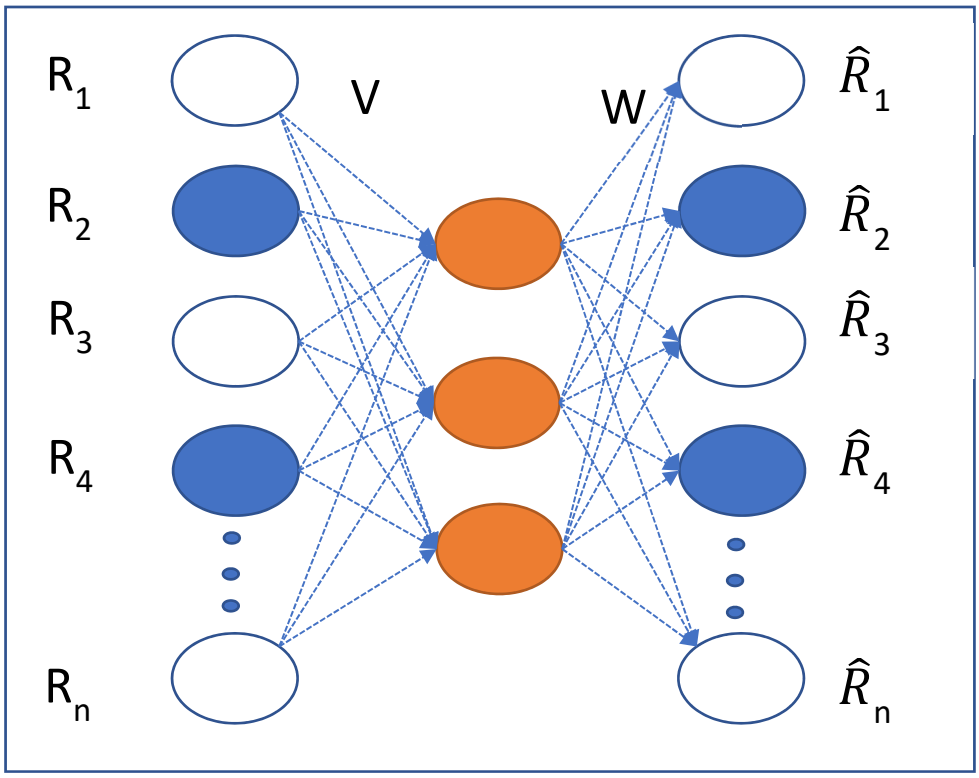

Figure 1. Structure of an autoencoder.

Heterogeneous Candidate Generation

Subsequently, another pioneering top-N recommendation system, the neural probabilistic language model [21], was proposed, which compresses sparse-valued information (one-hot vector) into a fixed real-valued embedding vector, $V^i = \left[v_1^i, \ldots, v_j^i \ldots, v_p^i\right] \in \mathrm{W}^{n \times p}$ (*W: embedding matrix, i: user id, p: size of the embedding vector*) (Figure 2). This vector learns a distributed representation for each word and the probability function of the sequence in a sentence.

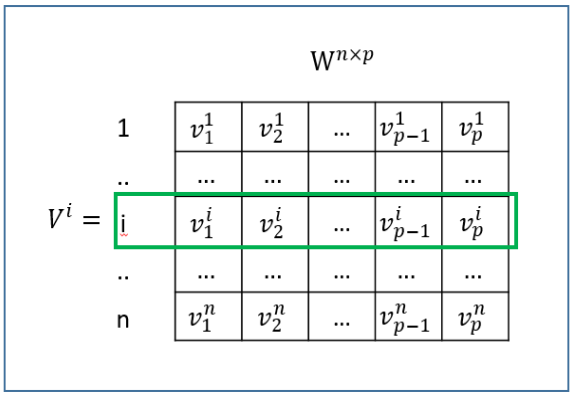

Figure 2. Embedding method.

The YouTube deep-recommendation system [10] is another example of candidate generation. It consists of two main parts: candidate generation and item ranking. The candidate-generation process reduces a large number of items to several hundreds by filtering out unrelated videos. The first layer of the candidate-generation neural network emulates this factorization algorithm without deep layers and only relies on the history of the users. This candidate-generation process can be considered a nonlinear version of the

factorization algorithm. The YouTube deep-recommendation model is most relevant to our proposed model. However, most recommendation systems do not consider the order of items. For example, in Bayesian personalized [22] ranking, the user's history is modified to a rating table that cannot save the order information of the user's history. The YouTube deep-recommendation system loses the sequence information by calculating the average of the embedded user history.

The proposed model utilizes the sequence data of user history to improve the prediction accuracy of deep-recommendation systems. The proposed model differs from other existing works in terms of the ability to change the users' preferences over time. However, to the best of our knowledge, no model that seamlessly combines heterogeneous information and sequential data simultaneously is currently available.

3. Proposed Methodology

This section illustrates our proposed deep-sequence–aware recommendation model that predicts candidates by integrating user history as sequence data and content information simultaneously. The following subsections detail the structure and role of each component of the proposed deep-sequence–aware recommendation model.

3.1. Sequence-Aware Embedding

Embedding involves mapping a discrete categorical variable to a vector of continuous numbers. In the context of neural networks, embeddings are low-dimensional learned continuous vector representations of discrete variables. Neural network embeddings are useful because they can reduce the dimensionality of categorical variables and meaningfully represent categories in the transformed space. As illustrated in Figure 3, embeddings are the weights of neural networks whose input is a one-hot encoding of categorical data. Where, h is node in hidden layer, p is size of embedding vector (hidden layer), V is embedding vector for item 3, v is a weight in embedding vector, N-number of items.

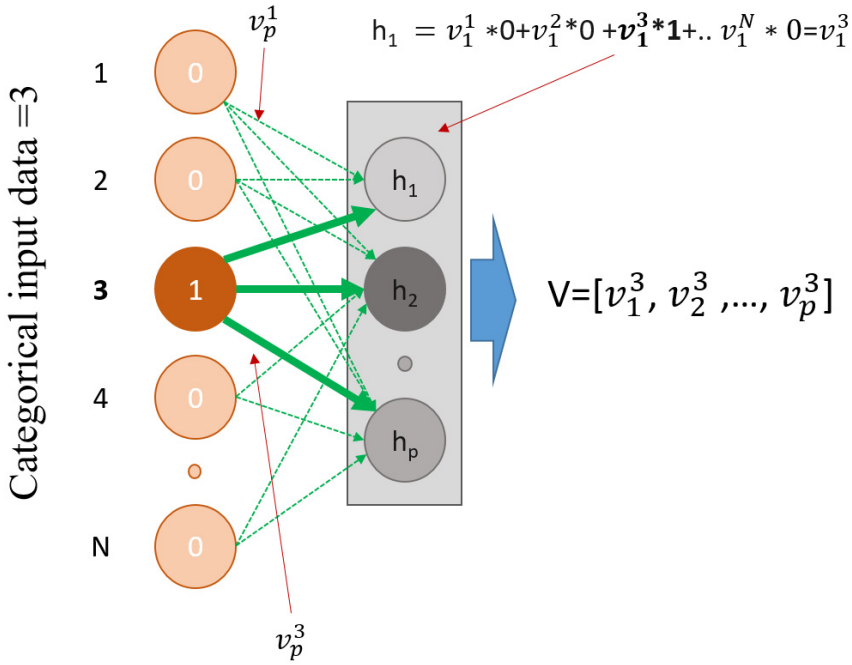

Figure 3. Example of calculating the embedding for item = 3.

To avoid computational costs, these weights are initialized as a table of variables (Figure 2). During the training process, the corresponding row of the table inputs the categorical values to the network, and then the variables of this row are trained, along with all the other weights in the network. Depending on situation, pre trained or random initialized weights can be used as embedding vector. In case of having lack of time, using

trained weights is recommended. Trained embedding can be generated by using the latent vector of traditional methods such as the BPR [22] model.

Because all fields of the Ubob.com dataset comprise categorical information and all fields have various lengths, analyzing data by traditional methods is difficult. In our case, the embedding layer is the number of courses that are different for each user. To solve this problem, we created a mask using the sequence length of user history. Then, we filtered out the useless embedding by multiplying a mask and an embedding vector. In a typical deep-recommendation system, the average vector of courses is first calculated using Equation (5):

$$\frac{1}{n}\sum_{i=0}^{n} V_i \tag{5}$$

where n is the length of the user's history, and V_i is the embedding of the item selected by the user. User preference is considered unchangeable during the time, and based on this embedding method, the same predictions are made for all combinations of the history of the members. However, this assumption may not be appropriate for all situations. First, we formulated a user–item relation table with $R \in \mathbb{R}^{m \times n}$, where m and n are the numbers of users and items, respectively. Every user $u \in U = \{1, 2, \ldots \mathrm{m}\}$ has $r^{(u)} = (R_{u1}, \ldots R_{um})$ history, and the history of each item $i \in I = \{1, 2, \ldots \mathrm{m}\}$ can be represented by $r^{(i)} = (R_{1i}, \ldots R_{mi})$. Second, we assume that for an e-learning system, the following data are given: one year ago, u_k took a math class, and now is studying a history class, whereas u_l finished the history class one year ago and has recently started the math class.

u_k: *{math, history}*

u_l: *{history, math}*

The preference for u_k is significantly different from that of u_l, who studied history and math sequentially. The first user may have already finished math, and his preference changes for the history class with time, whereas the second user wants to take the math course. This shows that the sequence is ignored, and recommending the same items may not be an accurate prediction.

To utilize sequence data (user history), we propose a new embedding vector that stores sequence information. To develop this model, the categorical history of the user should be input to the model without changing the order (such as (i_1, i_2, i_3 … i_N)). Because N varies for different users and neural networks can use a fixed matrix shape, the length of the history should be made equal by adding a certain constant number that will be removed in the next layer. In the next layer, this integer number (users' history) is input to the embedding layer, which replaces integers with the corresponding float dense vector. Commonly, a vector length of 128 is used for big datasets; however, it can be changed. It is important that the constant number that is added to make the length of the vector constant be removed by masking methods.

As mentioned earlier, items that were consumed a long time ago lose their importance with time, and items that have recently been consumed by users may have higher importance in the prediction of the next item. Thus, each embedding vector of user history is multiplied by values that explain the order of the item in the user's history (6).

$$V' = \frac{V}{N - order + 1} \tag{6}$$

Here, **order** is the order of items by time, which starts from 0; V is the embedding vector of the current item; and N is the number of items in the user's history. After calculating the sequential embedding of items, the average of these vectors is calculated using Equation (7).

$$V_{sequential} = \frac{1}{N}\sum_{i=1}^{N} V\prime_i \tag{7}$$

Then, as illustrated in Figure 4, $V_{sequential}$ is concatenated to the input layer, together with the actual embedding to inform the layer about both the sequential and average preferences of the user.

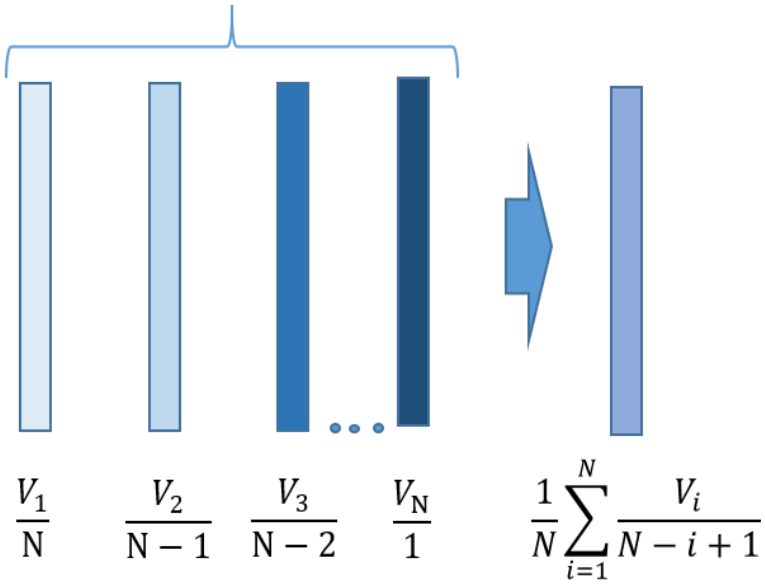

Figure 4. Sequence-based embedding.

3.2. Content Data

Typically, the content part of user data comprises big categorical data, small categorical data, and continuous data. Continuous data are input to the network as usual. Big categorical data should be input as an embedding vector. Unless the categorical data are small, they can be input into a network as a sparse vector (one-hot encoding). For example, if the user data had a gender field, it could contain three categorical values: male, female, and missing data. In this case, these data should be converted to the following one-hot encoding:

Female: [1, 0, 0]

Male: [0, 1, 0]

Missed data: [0, 0, 1]

In our data, the "jobcode" information of the user was included, which explains the user's job by keywords. Because the number of job codes is more than 100, and one user may have several job codes, we input this information as an embedding vector and computed the average vector and concatenated deep network.

3.3. Deep Network

The deep part of the network starts by concatenating all feature vectors in one layer as illustrated in Figure 5. Based on the length of the training data, we can add two or three fully connected layers (three in all our experiments). The number of units in each hidden layer was 64, 32, and 16. For the embedding size of items, we used 32 units as the length of the embedding vector, whereas 16 units were used for the "jobcode" and "company information" of users. Next, sigmoid, leaky ReLu, and ReLu functions were tested as activation functions. ReLu (7) was found to be the most suitable for the network.

$$ReLU = \begin{cases} x: \ if \ (x > 0) \\ o: \ if \ (x < 0) \end{cases} \tag{8}$$

Finally, SoftMax (8) was used as the output layer in our network, which had the same number of active items in the dataset.

$$\text{SoftMax} = \sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{9}$$

Notably, the number of output units must be the same as the number of items in our dataset.

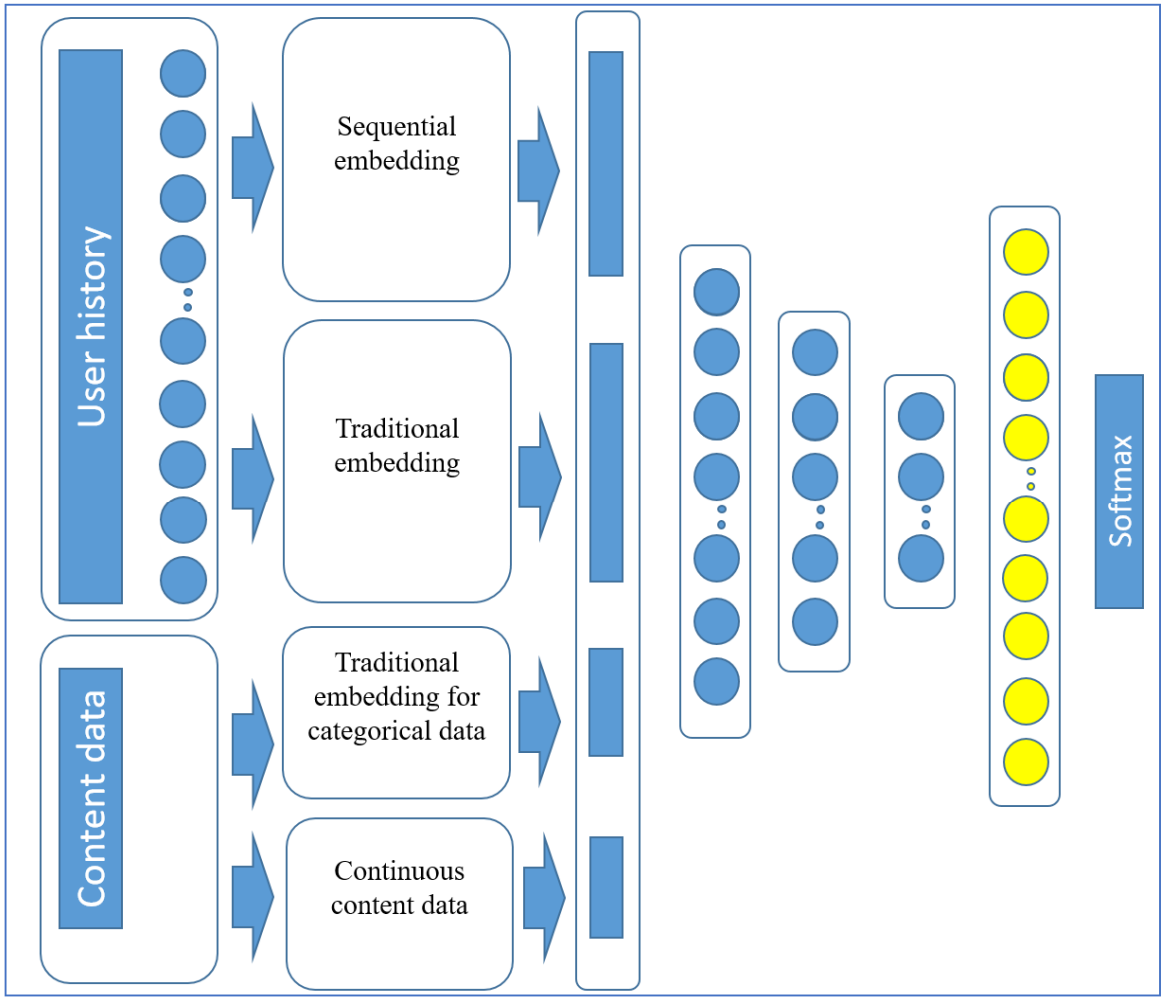

Figure 5. Structure of deep-sequence–aware recommendation.

3.4. Network Optimization

As the best solution for the optimization problem of the deep-sequence–aware network, stochastic gradient descent was selected as the optimization algorithm. Stochastic gradient descent can minimize the differentiable objective function iteratively by taking steps opposite to the gradient. The main difference between stochastic gradient descent and traditional gradient descent is that the stochastic gradient descent calculates the gradients among randomly sampled data. Because the gradient in gradient descent is calculated using all training data, the stochastic gradient descent is significantly faster than the gradient descent algorithm. During training, the optimal output was obtained when the learning rate η was equal to 0.00.

$$\theta = \theta - \eta \cdot \nabla_{\theta} J\left(\theta; x^{(i)}; y^{(i)}\right) \tag{10}$$

The pseudo code of the stochastic gradient descent algorithm is as follows:

- *Select an initial array of θ and learning rate η*
- *Repeat until an approximate minimum is calculated:*
- *Randomly shuffle examples in the training set:*
  - *For i = 1,2, . . . , n, do:*
    - $\theta = \theta - \eta \cdot \nabla_{\theta} J_i(\theta)$

4. Experimental Results

4.1. Experimental Setup

4.1.1. Datasets

Ubob.com Dataset

To build our recommendation system model and evaluate its performance, we mainly used the Korean Ubob.com e-learning dataset. As shown in the diagram below (Figure 6), the database has four tables that we used as heterogeneous and sequence data to utilize our recommendation system. All fields of the Ubob.com dataset consist of categorical information, and the lengths of fields vary. Thus, traditional methods cannot easily analyze these data. The most important fields are explained here:

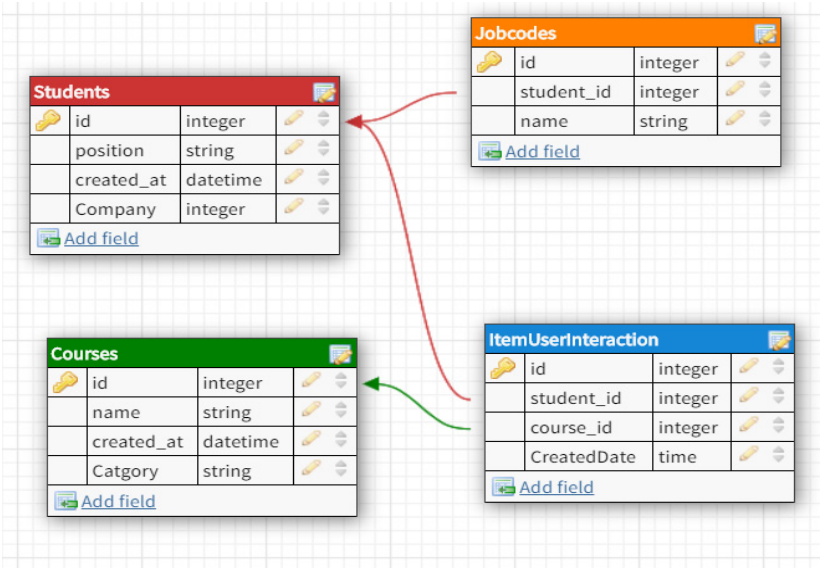

Figure 6. Structure of the Ubob.com dataset.

A student table was built to store user information. In most fields such as age and gender, the information was not sufficient to be used in recommender systems. Thus, these values were removed from the table. After removing all unusable values, we saved only the "jobcodes," course history, and item–user interaction tables. The number of all members in the database was approximately 222,000; however, only 82,000 users had item–user interaction data that can be used in our model. Every user has their own company information. In addition, students have job code information that details the profession of the student in one word. The database has a different number of job codes for each user. The overall number of valid job codes was 528. We had 82,000 users. If all data are trained using only the existing data, the last layer of the network (SoftMax) cannot learn all the classes. To tackle this issue, the amount of data was increased by a data-augmentation method, called the sliding window, which is explained next. If the history of one user is $\mathbf{I_1}$, $\mathbf{I_2}$, $\mathbf{I_3}$, and $\mathbf{I_4}$, this means that they selected $\mathbf{I_4}$ when they had history $\mathbf{I_1}$, $\mathbf{I_2}$, and $\mathbf{I_3}$, and when they had history $\mathbf{I_1}$ and $\mathbf{I_2}$, they selected $\mathbf{I_3}$, so we created the following data and increased the data size.

$X1 = [I_1, I_2, I_3]$ $Y1 = [I_4]$
$X2 = [I_1, I_2]$ $Y2 = [I_3]$
$X3 = [I_1]$ $Y3 = [I_2]$

After augmenting all the data, we obtained 228,000 data points that could be analyzed by the neural network. Eighty percent augmented data were used for training, whereas the remaining 20% were utilized as test data.

MovieLens 20M Dataset

This dataset contains 20,000,263 ratings and 465,564 tags applied to 27,278 movies by 138,493 users of the online movie recommender service MovieLens. Each user rates a movie from 1 to 5, where 1 is the worst and 5 is the best. Users were randomly selected for inclusion. All users selected had rated at least 20 movies. The dataset was randomly divided into 80% for training and 20% for testing.

4.1.2. Implementation Environment

In the experimental evaluation, the proposed and compared methods were deployed based on the configured hardware and software environments presented in Table 1.

Table 1. Parameters of experimental environment.

Classification	Details
OS	Windows 10
Programming Language	Python (Keras-Anaconda), TensorFlow
CPU	Intel(R) core (TM) Core i7-7700
RAM	16.00 GB

4.2. Performance of the Proposed Model

4.2.1. Training

In this section, we assess the performance of our method using the TensorFlow deep-learning framework. Data were divided into training and testing sets (80% and 20%, respectively). After training our model, we achieved a prediction accuracy of 60% and testing loss of 0.139. The visual representation of the loss functions of the training and validation sets shows that the network is trained without overfitting. In addition, because the testing loss started to increase (Figure 7), we can conclude that in the 20th epoch, the training model delivered the best performance.

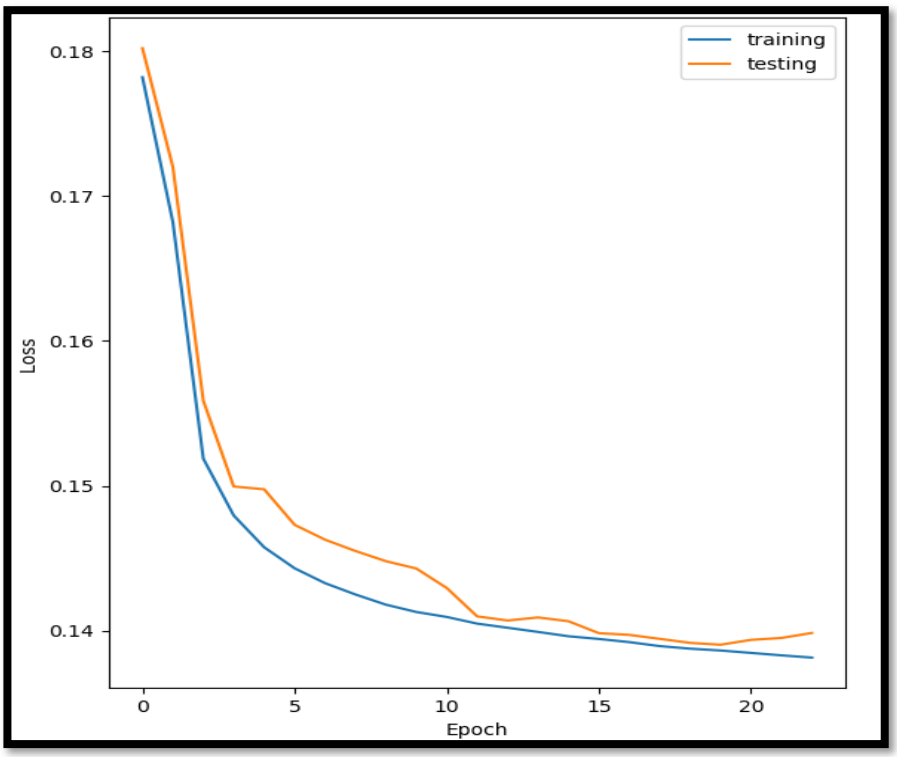

Figure 7. Training and testing losses in 20 epochs.

Using a trained model, we used the embedding of courses in latent space using principal component analysis (PCA) and searched the course called "Real Estate Broker_Introduction to Real Estate Science 2017(1)". The obtained answers are presented in Table 2, which we believe are worth using as k-NN.

Table 2. Similarity between courses.

Course Name	Similarity
Real Estate Broker_Civil Act and Special Civil Act)	0.737
Real Estate Broker_Real Estate Disclosure Act	0.697
Real Estate speciality	0.549
Real Estate Agent_Real Estate Act	0.533

4.2.2. Prediction Accuracy

Notably, the proposed deep-sequence–aware candidate-generation model completed the entire workflow of candidate generation by integrating user history as sequence data and content information simultaneously. To test the superiority of our model, four methods were selected for comparison, and the results are summarized in Table 3.

1. YouTube model: Proposed by Covington et al. [10], this method uses continuous and categorical features to recommend items by ignoring the sequence data of user history.
2. AutoRec: The model [20] attempts to exploit user preference data for items to provide personalized recommendations based on the autoencoder concept.
3. Deep interest network: Developed by Zhou et al. [23], deep networks map the historical behaviors of users into low-dimensional embedding vectors to learn nonlinear relations among features to predict the click-through rate.
4. DDPG (Tag-aware) model: Designed by Zhiruo Zhao et al. [24], the model is a tag-aware recommender system based on deep reinforcement learning without complex function design, taking advantage of tags to make up for the interpretability problems existing in the recommender system.

Table 3. Performance prediction.

Model	Ubob.Com Dataset			MovieLens 20M
	Precision 1	Precision 5	Precision 10	MAE
YouTube model	0.593	0.461	0.294	0.6471
Proposed deep-sequence–aware model	**0.599**	**0.482**	**0.358**	**0.6328**
AutoRec	0.309	0.213	0.175	0.6854
Deep interest network	0.32	0.254	0.197	0.7086
DDPG (Tag-aware)	-	-	-	0.7143

4.2.3. Dealing with the User Cold-Start Issue

One of the biggest challenges in building a recommender system is the cold-start problem that occurs when a new user or item is introduced, for which no past interactions are available. This is an important problem that is a well-researched issue.

The Ubob.com dataset has sufficient interaction information regarding users and items; thus, the new community issue can be avoided. In addition, courses are not seasonal and changeable; thus, new items are also not a problem for this e-learning system. Only making predictions for new users can be a problem.

To solve the problem of new users, during training and testing, we input a default embedding as user history to the network, together with their "jobcode" and "company id," so that the system learns predicting items based only on data that are usually available for new users. The comparative results are presented in Table 4, which show that the proposed method outperformed the other models. Thus, user history data as sequence information and content information can be used simultaneously to provide recommendations for

cold-start items. In both instances, clearly, the proposed methodology helped mitigate the cold-start problem for new users significantly better than the other models.

Table 4. User cold-start performance.

Model	Ubob.Com Dataset		MovieLens 20M
	MAE of 25 Cold-Start Users	MAE of 50 Cold-Start Users	MAE of 50 Cold-Start Users
YouTube model	0.651	0.721	0.6987
Proposed deep-sequence–aware model	0.637	0.685	0.6901
AutoRec	0.753	0.802	0.7458
Deep interest network	0.749	0.787	0.7136
DDPG (Tag-aware)	-	-	0.7044

5. Conclusions

Although the development of recommender systems is progressing at a high rate, user- and item-cold-start problems and the accuracy of recommender systems are open challenges that must be overcome. Herein, we proposed a novel method for recommendation systems that simultaneously integrates content (heterogeneous) information and sequence data. In addition, by utilizing default embedding as nonexistent history, our model can provide recommendations for users who do not have item interaction data. The proposed model differs from other existing works in terms of the ability to identify changes in user preferences over time. In addition, the experimental results revealed the significant influence of the combination of sequence and content (heterogeneous) data on alleviating the user-cold-start problem; in terms of the accuracy of the recommendation system, our proposed model was superior to other models such as the collaborative autoencoder, deep interest network, and deep YouTube recommendation model. Our model performed well on the Ubob.com dataset, which is highly dependent on sequence data. Despite the superiority of the proposed approach, several problems were encountered. In particular, with advances in the domain, a high volume of data is available for making recommendations. Therefore, future research could explore more sophisticated models to estimate the importance of the hidden features of users and items that are represented as a sequence as well as content (heterogeneous) information preference using recent deep-learning methods and algorithms. Additionally, in future research, an explainable and interpretable recommendation system based on sequence and content features could be developed.

Author Contributions: This manuscript was designed and written and the experiments were performed by A.I.; A.K. helped revise and improve the model and the manuscript; the theory and experiments were analyzed and commented on by T.-K.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the GRRC program of Gyeonggi province. [GRRC-Gachon2021 (B04), Development of AI-based Healthcare Devices].

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Acknowledgments: A.I and A.K. would like to express their sincere gratitude and appreciation to their supervisor, Taeg Keun Whangbo (Gachon University), for his support, comments, remarks, and engagement over the period in which this manuscript was written. Moreover, the authors would like to thank the editor and anonymous referees for their constructive comments on improving the content and presentation of this paper.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Kutlimuratov, A.; Abdusalomov, A.; Whangbo, T.K. Evolving Hierarchical and Tag Information Via the Deeply Enhanced Weighted Non-Negative Matrix Factorization of Rating Predictions. *Symmetry* **2020**, *12*, 1930. [CrossRef]
2. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
3. Fang, Y.; Si, L. Matrix co-factorization for recommendation with rich side information and implicit feedback. In Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 65–69.
4. Ricci, F.; Rokach, L.; Shapira, B.; Kantor, P.B. *Recommender Systems Handbook*; Springer: Berlin, Germany, 2011; ISBN 978-0-387-85819-7.
5. Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. Recommender systems survey. *Knowl. Based Syst.* **2013**, *46*, 109–132. [CrossRef]
6. Wang, J.; de Vries, A.P.; Reinders, M.J.T. Unifying User-based and Item-based Collaborative Filtering Approaches by Similarity Fusion. In Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, WA, USA, 6–11 August 2006; pp. 501–508.
7. Su, X.; Khoshgoftaar, T.M. A survey of collaborative filtering techniques. *Adv. Artif. Intell.* **2009**, *2009*, 1–19. [CrossRef]
8. Zheng, L.; Lu, C.-T.; He, L.; Xie, S.; He, H.; Li, C.; Noroozi, V.; Dong, B.; Yu, P.S. MARS: Memory Attention-Aware Recommender System. In Proceedings of the 2019 IEEE International Conference on Data Science and Advanced Analytics (DSAA), Washington, DC, USA, 5–8 October 2019; pp. 11–20.
9. Ruining, H.; Julian, J. VBPR: Visual bayesian personalized ranking from implicit feedback. In Proceedings of the AAAI-16 Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
10. Covington, P.; Adams, J.; Sargin, E. Deep neural networks for youtube recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 191–198.
11. Deshpande, M.; Karypis, G. Item-based top-N recommendation algorithms. *ACM Trans. Inf. Syst.* **2004**, *22*, 143–177. [CrossRef]
12. Schein, A.I.; Popescul, A.; Ungar, L.H.; Pennock, D.M. Methods and metrics for cold-start recommendations. In Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, New York, NY, USA, 11–15 August 2002; pp. 253–260.
13. Lam, X.N.; Vu, T.; Le, T.D.; Duong, A.D. Addressing cold-start problem in recommendation systems. In Proceedings of the 2nd International Conference on Ubiquitous Information Management and Communication, Suwon, Korea, 31 January–1 February 2008; ACM: New York, NY, USA, 2008; pp. 208–211.
14. Agrawal, R.; Srikant, R. Mining sequential patterns. In Proceedings of the 11th International Conference on Data Engineering (ICDE), Taipei, Taiwan, 6–10 March 1995; pp. 3–14.
15. Mobasher, B.; Dai, H.; Luo, T.; Nakagawa, M. Using sequential and non-sequential patterns in predictive Web usage mining tasks. In Proceedings of the 2002 IEEE International Conference on Data Mining, Maebashi City, Japan, 9–12 December 2002; pp. 669–672. [CrossRef]
16. Yu, H.; Riedl, M.O. A sequential recommendation approach for interactive personalized story generation. In Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems, Valencia, Spain, 4–8 June 2012.
17. Zhao, G.; Lee, M.L.; Hsu, W.; Chen, W. Increasing temporal diversity with purchase intervals. In Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, Portland, OR, USA, 12–16 August 2012.
18. Qiao, Z.; Zhang, P.; Cao, Y.; Zhou, C.; Guo, L.; Fang, B. Combining Heterogenous Social and Geographical Information for Event Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014; pp. 165–174. [CrossRef]
19. Bao, Y.; Fang, H.; Zhang, J. TopicMF: Simultaneously Exploiting Ratings and Reviews for Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Québec City, QC, Canada, 27–31 July 2014.
20. Sedhain, S.; Menon, A.K.; Sanner, S.; Xie, L. AutoRec: Autoencoders Meet Collaborative Filtering. In Proceedings of the 24th International Conference on World Wide Web, Florence, Italy, 18–22 May 2015; pp. 111–112.
21. Bengio, Y.; Ducharme, R.; Vincent, P.; Janvin, C. A Neural Probabilistic Language Model. *J. Mach. Learn. Res.* **2003**, *3*, 1137–1155.
22. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian Personalized Ranking from Implicit Feedback. *arXiv* **2012**, arXiv:1205.2618.
23. Zhou, G.; Zhu, X.; Song, C.; Fan, Y.; Zhu, H.; Ma, X.; Yan, Y.; Jin, J.; Li, H.; Gai, K. Deep Interest Network for Click-Through Rate Prediction. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 1059–1068. [CrossRef]
24. Zhao, Z.; Chen, X.; Xu, Z.; Cao, L. Tag-Aware Recommender System Based on Deep Reinforcement Learning. *Math. Probl. Eng.* **2021**, *2021*, 5564234. [CrossRef]