applied sciences

MDPI

Article

Deep Learning-Based Context-Aware Recommender System Considering Contextual Features

Soo-Yeon Jeong and Young-Kuk Kim *

Department of Computer Science & Engineering, Chungnam National University, Daejeon 34134, Korea; susan92@cnu.ac.kr
* Correspondence: ykim@cnu.ac.kr

Abstract: A context-aware recommender system can make recommendations to users by considering contextual information such as time and place, not only the scores assigned to items by users. However, as a user preferences matrix is expanded in a multidimensional matrix, data sparsity is maximized. In this paper, we propose a deep learning-based context-aware recommender system that considers the contextual features. Based on existing deep learning models, we combine a neural network and autoencoder to extract characteristics and predict scores in the process of restoring input data. The newly proposed model is able to easily reflect various type of contextual information and predicts user preferences by considering the feature of user, item and context. The experimental results confirm that the proposed method is mostly superior to the existing method in all datasets. Also, for the dataset with data sparsity problem, it was confirmed that the performance of the proposed method is higher than that of existing methods. The proposed method has higher precision by 0.01–0.05 than other recommender systems in a dataset with many context dimensions. And it showed good performance with a high precision of 0.03 to 0.09 in a small dimensional dataset.

Keywords: recommender systems; context-aware; deep learning; autoencoder; neural network

Citation: Jeong, S.-Y.; Kim, Y.-K. Deep Learning-Based Context-Aware Recommender System Considering Contextual Features. *Appl. Sci.* **2022**, *12*, 45. https://doi.org/10.3390/app12010045

Academic Editors: Vassilis Charissis and Dimitris Drikakis

Received: 20 September 2021
Accepted: 18 December 2021
Published: 21 December 2021

1. Introduction

A recommender system is used to give recommendations to allow users who desire information to quickly find such information when given lots of information. When e-commerce services were newly emerging, users had to invest a significant amount of time to find what they wanted from lists of products or services. However, as recommender systems became available, and users found the technology very useful, and the services were able to increase their revenues. At present, recommender systems are adopted by e-commerce services in various fields such as Netflix, Amazon, and YouTube. Many ongoing researches are exploring how to apply them to each field [1–3].

Before smartphones became widely available, recommender systems were mainly applied to accessed on desktop computers. They operated based upon basic information—users' purchase history or evaluation scores—and in the early days, recommender systems were modeled by focusing only on users and items only. Considering that contextual factors can impact users' preferences when they select an item, such information can further improve the performance of such systems. For instance, if users watch a movie alone, then it is acceptable to apply the existing recommender system, but if they watch it with their kids, then it is more likely that they will choose to watch G-rated or animated movies in the future. When recommending music to users at home, focus can be placed on either the latest music or their favorite music. However, if users listen to music when they are near the sea, then exciting music or music related to the sea could be recommended. As mentioned earlier, depending on whom they are with or where they are, their preference can be changed. This is called a context-aware recommender system [4], and it is assumed that users' specific context can affect their preferences. At present, contextual information can be collected from users' smartphones or Internet of Things (IoT). The main types of

contextual information that are used include time, weather, and place. Depending on the recommended content, traffic conditions or companion information can be also used [5,6]. Categories like weather and location are referred to as contextual dimensions, and information such as sunny, rainy, or gloomy weather, or presence at home or a movie theatre, are called contextual conditions [7]. For a context-aware recommender system, we can use smartphones or IoT to gather various contextual information, but the more contextual information we consider to predict preferences, the more severe data sparsity becomes. In addition, if there is no data regarding a user's context, a context-aware recommender system makes a recommendation based on similar users' contextual data. However, it is difficult to find reliable similar users in the case of data sparsity. The method to solve this problem is matrix factorization (MF), which shows high performance in cases of sparse data [8–12]. MF finds the factors that are assumed to potentially exist in the data based on the latent factor model. It is highly effective for predicting empty values that are not evaluated in the user/Item matrix.

Due to recent exponential increases in the amount of data that can be utilized, as well as with rapid developments in PC hardware, deep learning has been applied to various fields with excellent results [13–16]. Several ongoing researches are exploring how to apply deep learning to recommender systems. More specifically, some researches are exploring how to improve performance by integrating the main feature of existing recommender systems, collaborative filtering, with deep learning [17–19]. A deep learning-based recommender system has successfully improved an existing recommender system's performance and accuracy by extracting meaningful features from various data, such as images and review texts [20,21]. However, there is not yet sufficient research on a deep learning-based context-aware recommender system. Since the amount of context type to consider keeps increasing, there is a need for a recommender system that can easily reflect contextual information while alleviating sparsity issues. When there are similar users or items, there can be similar contextual information. In such cases, it is possible to improve the existing recommender system's performance by reflecting the context features. In this paper, we would like to propose a context-aware recommender system that integrates a neural network and an autoencoder. MF, which shows great performance in the existing context-aware recommender system cannot resolve nonlinear problem such as XOR. We can use deep learning to resolve issues that could not be classified in the past. In addition, to reflect contextual features other than user and item characteristics, an autoencoder is combined with a neural network, enabling us to improve the accuracy of prediction scores while extracting contextual features. The accuracy of performance evaluations will be measured using music, food, and movie recommendation dataset. For comparison purposes, the previously mentioned other models are subject to comparison.

The paper is organized as follows. Section 2 describes ongoing researches on the existing context-aware recommender system and a deep learning-based recommender system. Section 3 describes a deep learning-based context-aware recommender system that considers contextual features, which is a newly proposed model. In Section 4, the performance of newly proposed method and existing method will be evaluated. Finally, in Section 5, conclusions will be drawn, and future research direction will be discussed.

2. Related Works

Earlier, it was mentioned that a context-aware recommender system requires various type of contextual information. In the past, date and time information were mainly used because of their easy availability. Nowadays, it is possible to gather contextual information, such as place, time, and mood, from various users via IoT device sensors or smartphones. As the amount of contextual information to consider increases, it becomes possible to provide more detailed recommendations. At the same time, users are required to provide detailed contextual data, and it is highly unlikely that they will evaluate every item in every context. For this reason, in most cases, contextual data are sparse. To reduce the data sparsity problem, we can use matrix factorization, a latent factor model based collaborative

filtering [22,23]. matrix factorization can be used effectively to predict that scores of items not evaluated by users by finding the latent factors that exist between users and items. Since matrix including contextual data are multidimensional, tensor factorization (TF) is possible [24,25]. Although TF shows better performance than MF, due to its high complexity, one can apply the weights of contextual features and maintain the user/item matrix. In this model, multidimensional data, including context are expressed in a matrix. Item splitting, in which an item and the contextual condition are handled as one fictious item, is used to create a matrix [26,27]. However, there are many items and many contextual conditions for each item. Thus, due to complexity and data sparsity in the item splitting process, not every context is reflected. Factorization machine (FM) is more suitable when the dataset has many features, such as tags and genres [28]. FM can be used to insert many features into a model and then model the correlation among variables. Although it shows high performance even with scarce data, it has only one latent vector for one feature. Thus, assuming there are latent vectors A, B, and C, it is unable to distinguish between the impacts of A and B or A and C.

As deep learning has become more popular, researchers have begun to successfully apply deep learning models to recommender systems and utilize various data types to improve performance. NN applied Neural Collaborative Filtering is simple and effective network. To model interactions between users and items, a dual neural network has been constructed [29]. NCF is characterized by the use of a latent vector obtained by applying MF to user/item matrix. Autoencoder is an unsupervised learning algorithm with no label. The model is run to reduce the input data to optimal dimensions and restore the original data. The input and output of the model dimensions are same. An autoencoder's hidden layer is expressed by the characteristics of input data. The autoencoder is used for matrix completion, as it can fill empty data (i.e., those that used to be null) within the recommender system based on user/item characteristics. In addition, I-AutoRec and U-AutoRec can restore original inputs by using user/item rating as inputs [30]. Recently, there has been an increase in research on deep learning recommender systems that improve accuracy is improved by reflecting the impacts on the user/item using the additional information. Deep learning is useful for processing unstructured data such as text review data or image data. Compared to the existing recommender system, it is possible to handle a greater variety of data and assess the complex relationships among such data. However, there is a lack of current research on the characteristics of contextual information in the context-aware recommender system. In the past, recommendations were based on the impacts of context on items or users. In some cases, the ratings evaluated by users in a specific context are densely located near a certain rating. It can be high or low in the users' preference, depending on the other contexts. This means that there may be correlation between contextual condition. An autoencoder can be used to automatically find the correlation of contextual information without designating it in advance. There is a recommender system using the characteristics of users or items using AE, and there is a recommender system using AE for the characteristics of social networks of SNS [31,32]. Therefore, the proposed method utilizes AE to reflect the characteristics of the context. To predict scores based on the contextual features obtained from an autoencoder, the newly proposed model is trained by using scores as target while adjusting weights. This model in the research will be discussed in detail in Section 3.

3. Proposed Model

This section discusses the proposed model, a deep learning-based context-aware recommender system that integrates an autoencoder and neural network. First, a general Autoencoder used to reveal the characteristics of context feature is explained. After that, the proposed model is presented and its learning process are discussed.

3.1. Learning Context Using Autoencoder

In most cases, in a recommender system, collaborative filtering is classified into user-based CF and item-based CF. After measuring each user's (item's) similarity, user (item) correlations are assessed to make a proper recommendation. In prior research on recommender systems, relatively limited focus was placed on the contextual feature. Additionally, the contextual information applied to recent recommender systems mainly concerns time and weather. In this paper, we aim to reflect contextual features and much more diverse information by utilizing autoencoder. Within a recommender system, an autoencoder is mainly used to reduce the dimension of input data or extract the characteristics of input data. Focus is placed on the latter in the autoencoder used in the proposed model. Figure 1 illustrates a general autoencoder model in which context is used as an input data.

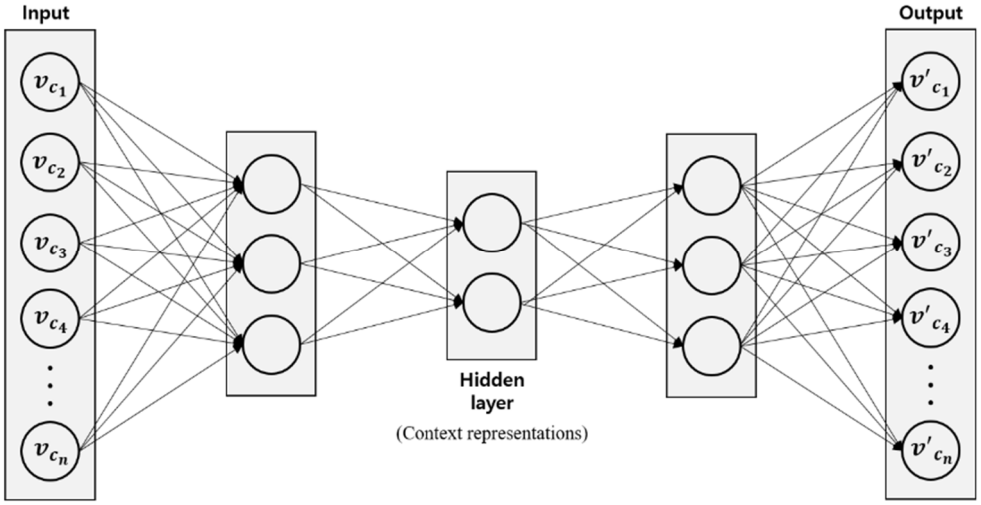


Figure 1. Context based Autoencoder.

In Figure 1, according to the left input, the contextual dimension is composed of vector v_{c_n}, where n is the number of contextual dimensions. The more layers that are present, the higher the accuracy. However, the presence of more layers does not always result in better performance; experimentation is required to set the proper number of layers. The formulas used to train autoencoder and minimize errors are described in the next sub-section.

$$\min_{\theta} \sum_{v \in V} \|X - X'\|^2 \tag{1}$$

X indicates input data. Input data X is passed through AE and comes out as the output X'. Formula (1) indicates learning while reducing errors between X and X'. $\theta = \{w, b\}$ indicates the model parameters. kth layer output x^k can be formulated as follows.

$$x^k = \sigma\left(w^k x^{k-1} + b^k\right) \tag{2}$$

w^k represents weight and b^k represents bias. $\sigma(x)$ is the activation function and we applied Exponential Linear Unit (ELU). ELU is the solution for dying ReLU problems where 0 is outputted when it has a negative value.

$$\sigma(x) = \begin{cases} x & if \ \ x > 0 \\ \alpha(e^x - 1) & if \ \ x \leq 0 \end{cases} \tag{3}$$

In Figure 1, the hidden layer in the middle represents the context when the input data is context. Focusing on this, parts of the newly proposed model in Section 3.2 will be learned.

3.2. Deep Learning

Basically, our motivation is to capture the contextual features to improve the recommender system with neural networks. The newly proposed framework is illustrated in Figure 2.

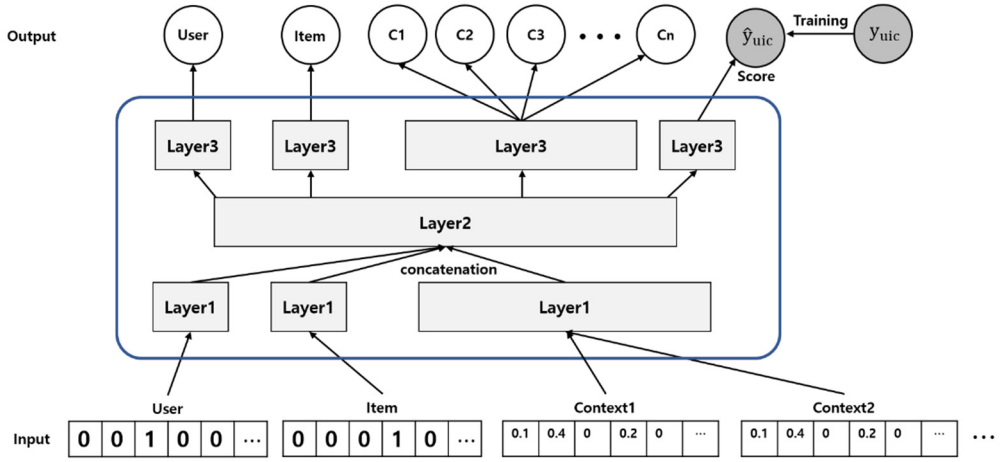


Figure 2. Neural Context-aware Recommender Systems.

The proposed model is formed by connecting multiple AEs and NN. The input data include multiple feature vectors. The input and output data for the autoencoder (discussed in Section 3.1) include user vectors, item vectors and context vectors. The neural network is trained by using score as targets from the hidden layer in the middle. When the dataset has n (x, y) pairs, x represents the data record containing users, items and contextual dimension, and y represents the score given by the user. Each field of x is expressed as a binary vector with one-hot encoding. This is similar to that of factorization machine. In Figure 2, y_{uic} is the actual score obtained from the evaluation of item i by user u in the context c, while $\hat{y}_{uic}$ is the predicted score.

$$\hat{y}_{uic} = \sigma\left(wx^{k/2} + b\right) \tag{4}$$

Assuming the total number of layers is k, then $\hat{y}_{uic}$ can be obtained from layer $k/2$. Also, $x^{k/2}$ can be obtained from formula (2). To train the proposed model, there are two processes one must go through. First, as shown in Figure 2, it must minimize the difference between output's $v_u{}'$, $v_i{}'$, and $v_c{}'$ and input's v_u, v_i and v_c where context which is the input data is separated as the context dimension. This process can be illustrated by formula (5).

$$L_{AE} = \min_{\theta} \sum_{v \in V} \|X - X'\|^2 + \alpha \sum_{l} \left(\|w_{AE}\|^2 + \|b_{AE}\|^2 \right) \tag{5}$$

To prevent overfitting, regularization terms are added in Formula (1).

Second, using the feature obtained from formula (5), $\hat{y}_{uic}$ is obtained. Training is performed to minimize the difference between y_{uic} and $\hat{y}_{uic}$, as expressed as Formula (6).

$$L_{NN} = \min_{\theta} \sum_{y \in Y} (y_{uic} - \hat{y}_{uic})^2 + \beta \sum_{l} \left(\|w_{NN}\|^2 + \|b_{NN}\|^2 \right) \tag{6}$$

The newly proposed model learns by applying Formulas (5) and (6) at the same time as well as Adaptive Moment Estimation (Adam). Adam is one of the most frequently used optimization methods. The Algorithm 1 is summarized based on the above derivation.

Algorithm 1 Algorithm of Proposed Model

Input: User vector U, Item vector I, Context vector C_n, Learning rate N, number of iterations k
Output: parameter w_{AE}, w_{NN}, b_{AE}, b_{NN}, Rating vector Y
Randomly initialize parameters w_{AE}, w_{NN}, b_{AE}, b_{NN}
for $i = 1$ to k **do**
for $u \in U$ **do**
Update w_{AE}, b_{AE} using $L_{AE}{}'$
Update w_{NN}, b_{NN} using $L_{NN}{}'$
end for
end for

It is necessary to verify whether our newly proposed method, which includes both autoencoder and neural network, is better than the existing method, which using either autoencoder or neural network. Therefore, to compare the proposed and existing model, Section 4 measures the performance of a context-aware recommender system that uses neural network and compares it to the context-aware recommender system that excludes the steps in Formula (5). The input data is those shown in Figure 2. The only output data is $\hat{y}_{uic}$, which is learned by using Formula (6).

4. Experiments

4.1. Datasets

To test the newly proposed method, a total of 3 datasets in Table 1 are used. The datasets are provided by CARSKit [33], and DePaulMovie, InCarMusic and Res-taurant (Tijuana) were used.

Table 1. Context-aware Datasets.

	Movie	**Music**	**Restaurant**
#of users	123	42	50
#of items	79	139	41
#of ratings	5043	4013	896
Dimension	3	8	1
Rating Scale	1–5	0–5	1–5
Data sparsity	94.5	99.9	93.4

- DePaulMovie dataset has a total of 3 contextual dimensions—Time, Location and Companion. Time has a total of 2 contextual conditions—Weekend and Weekday. Location has a total of 2 contextual conditions—Home and Cinema. Companion has a total of 3 contextual conditions—Alone, Family and Partner. DePaulMovie has the smallest set of context features.
- InCarMusic dataset has a total of 8 contextual dimensions—Driving Style, Road type, Landscape, Sleepiness, Traffic conditions, Mood, Weather and Natural Phenomena. In total, there exist a total of 26 contextual conditions.
- Restaurant (Tijuana) dataset is formed by combining time (Weekday, Weekend) and location(School, Home, Work).

5-fold cross validation is performed, 80% of users are used as training sets while 20% are used as test sets.

4.2. Evaluation Measures

There are many types of evaluation measures for evaluating the performance of a recommender system. For example, there are RMSE (Root Mean Square Error) and MAE (Mean Average Error) that are used to measure the errors of predicted scores. Methods for making a recommendation based on preferences, not scores, include the precision and recall methods. The dataset is expressed as scores, not preferences, but in the actual recommender system, users do not see scores. Instead, based on the ranking of predicted scores, the best

item is recommended. To measure performance based on ranking, the precision and MAP (Mean Average Precision) methods are used. *Precision@N* is shown as follows.

$$Precision@N = \frac{|\{relevant\ items\} \cap \{top-N\ items\}|}{N} \tag{7}$$

Precision represents the ratio of the items in which the user is interested to the recommended items. The list of items recommended based on the <User, Context> pair is provided to measure the hit ratio. To determine a preference, a score of 4 or high is set as the threshold. *relevant items* means items with a rated score of 4 or more, and items common to *relevant items* and $top-N$ *items* are included in the molecule. When recommending an item in a recommender system, which item is at the top of the list is critical. However, the precision measurement method does not take the order of appearance into account. MAP can be used to measure accuracy while considering the order of appearance. Before performing MAP, AP (Average Precision) should be obtained. *AP@N* can be obtained from Formula (8).

$$\begin{aligned} AP@N &= \tfrac{1}{m}\sum_{k=1}^{N}\Big(P(k)if\ k^{th}\ item\ was\ relevant\Big) \\ &= \tfrac{1}{m}\sum_{k=1}^{N}P(k)rel(k) \\ rel(k) &= \begin{cases} 1, & if\ relevant \\ 0, & otherwise \end{cases} \end{aligned} \tag{8}$$

In the above formula, m represents the number of relevant items, and *P(k)* is the precision value up to index k. In addition, *rel(k)* can be either 1 or 0 depending on whether the item was relevant or not. AP measures the accuracy of recommendations to one user, while MAP pertains to all users, U. *MAP@N* can be obtained via Formula (9). In the experiment for the newly proposed method, N is set to 10.

$$MAP@N = \frac{1}{|U|}\sum_{u=1}^{|U|}(AP@N)_u \tag{9}$$

4.3. Compared Methods

Non-contextual algorithms, User KNN, SVD++ and PMF (Probabilistic Matrix Factorization), are included. The neighbor of userKNN was set to 10, the number of factors was set to 50 for SVD, and 100 for PMF.

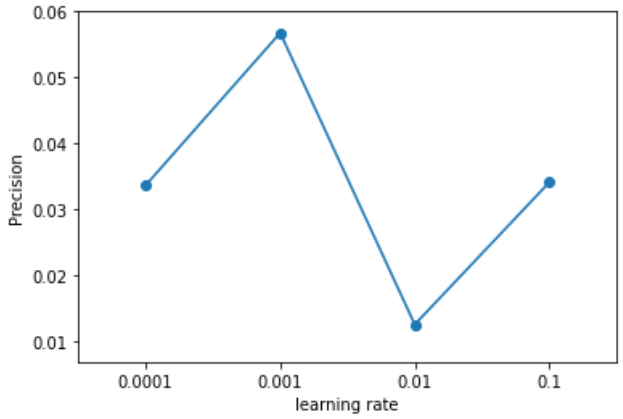
We selected CAMF (Context-aware Matrix Factorization), ItemSplitting-BiasdMF, and CSLIM (Contextual Sparse Linear Method) as state-of-the-art context-aware recommendations to compare with the proposed model. CAMF set the factor to 200 and ItemSplitting-BiasdMF and CSLIM to 100. FM set the factor to 300 and PMF to 50. In addition, it is necessary to verify whether a method combining AE and NN offers better performance than the existing deep learning method in terms of accuracy of predicting scores. For comparison purposes, NN that outputs score only without restoring input data is used.

4.4. Results of the Experiments

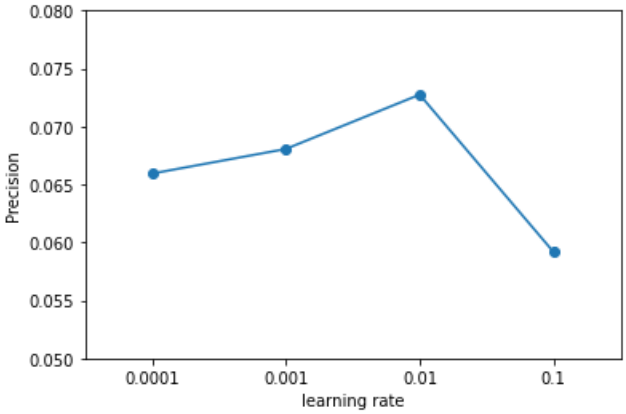
In the development environment, the operating system was windows10, RAM was 16G, the language was Python, and the development tool was a Jupyter notebook.

The following figure shows the experiment result to see the precision depending on learning rate.

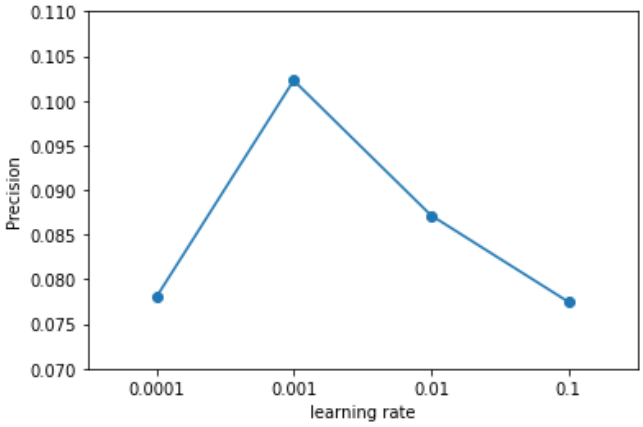
We use the Music, Restaurant, and Movie datasets. The results are shown in Figure 3. Although the best value is different for each dataset, most of them showed good results when the learning rate was 0.001.



(**a**) Music



(**b**) Restaurant



(**c**) Movie

Figure 3. Precision on Music, Restaurant, and Movie datasets.

The following table shows the test results for the music, restaurant, and movie datasets obtained from the proposed method and the compared method.

Tables 2 and 3 show the MAP and precision for the dataset obtained from the proposed method and the compared method. If the datasets are sorted in decreasing order of contextual dimensions, they are sorted as restaurant, movie and music. For the restaurant dataset, the non-contextual algorithm (i.e., in other words, the user KNN) yields similar results to those obtained from the proposed method. However, for the movie and music datasets, which have relatively higher contextual dimensions, the proposed method yields

much better performance than the user KNN. In addition, the proposed method for the restaurant dataset yields precision that are about 6% higher than the PMF. Similar to the user KNN, the proposed method yields better performance than the PMF for the movie and music datasets. For the restaurant dataset, the neural network-based method with deep learning technology and the proposed method yields better performance than other MF based methods, except for the PMF based method. The restaurant dataset has fewer contextual dimensions, so there is no significant difference in performance between the proposed method and other methods. However, the movie dataset has relatively higher contextual dimensions than the restaurant dataset, so for the movie dataset, the proposed method yields much better performance than other methods. Also, the music dataset with the highest contextual dimensions, shows much better performance than the other models. From this, we can conclude that for the dataset with the highest contextual dimensions, the proposed method yields much better performance than the existing methods. Finally, if the NN based method is compared to the proposed method, it seems that for every dataset, the proposed method yields higher precision and MAP. For the music dataset with high data sparsity, the proposed method yields about 17% higher precision than PMF's and shows about 4% higher MAP than ItemSplitting's. For the movie dataset, the proposed method yields about 5% higher precision than PMF's and shows about 2 times higher MAP than userKNN's. For the restaurant dataset, the proposed method yields about 6% higher precision than PMF's and shows about 8% higher MAP than ItemSplitting's. From these results, we can conclude that the proposed method is more effective than the context-aware recommender system in which deep learning-based techniques are applied.

Table 2. Performance comparison result in Precision.

Precision@10	Music	Restaurant	Movie
UserKNN	0.010082	0.062889	0.063732
SVD++	0.045452	0.062174	0.059719
CAMF_CI	0.024234	0.050423	0.058384
ItemSplitting-BiasedMF	0.045365	0.06249	0.054859
CSLIM_CI	0.003996	0.012763	0.004598
PMF	0.048151	0.068362	0.06807
FM	0.010663	0.047136	0.027956
NN based model	0.050387	0.059913	0.076811
Proposed model	**0.056760**	**0.072748**	**0.102263**

Table 3. Performance comparison result in MAP.

MAP@10	Music	Restaurant	Movie
UserKNN	0.036538	0.110237	0.107568
SVD++	0.171105	0.17123	0.070776
CAMF_CI	0.070767	0.121712	0.087723
ItemSplitting-BiasedMF	0.188662	0.175772	0.07685
CSLIM_CI	0.006741	0.066281	0.011318
PMF	0.179647	0.203266	0.095294
FM	0.037309	0.086927	0.073874
NN based model	0.147583	0.175486	0.224978
Proposed model	**0.196248**	**0.190286**	**0.268133**

The proposed method shows better performance than the existing recommender system in datasets with high data sparsity, so it can be said to be a supplementary model for the problem of data sparsity. Also, considering that the accuracy of the recommender system is high, it can be seen that it is effective for nonlinear data by using deep learning.

5. Conclusions

In the paper, we propose a deep learning-based context-aware recommender system for capture context feature. Also, a method for learning sparse data that employs autoencoder and neural network is proposed to overcome, the data sparsity problem faced by the existing context-aware recommender system. To learn contextual features, a method for integrating autoencoder for users, items and context is proposed. The model combines autoencoder, a non-supervised learning model that finds input characteristics by learning model and neural network, a supervised learning method that uses score output as targets. Users, items and context dimension are set as inputs and outputs so that the scores reflect each one's characteristics and relationships. This method can be easily applied when there is additional contextual information. The performance results of the proposed method show that it performs better on all datasets compared to most of the comparisons. In addition, it performs better than the comparison target for a dataset with a large amount of contextual information, since the proposed model further reduces the data sparsity problem. Because the proposed model is further reduced to the data sparsity problem. Also, the proposed model was better than only using the deep learning based neural network. It can be seen as a prediction of users' preference while capturing the context feature. Although the proposed method showed good results, there are some limitations. First, the proposed method has a slower response time than the existing recommender system. It was faster than user KNN, which measures similarity with FM, but slower than other methods. Second, an experiment was conducted using a data set suitable for the purpose of this paper, but the amount of data was small. In the future, we plan to conduct experiments by selecting a larger dataset.

In future research, efforts will be made to study a deep learning context-aware recommender system that reflects reliability. At present, the deep learning recommender system shows good performance, but it is impossible to interpret the results. In the case of a deep learning-based context-aware recommender system, it is impossible to know what kind of context affects the user's preference. The existing recommender system measures the reliability of the recommendation list (number# of similar users and data distribution/concentration), reduces the score of an item with lower reliability even if it received a high score, and recommends items with higher reliability and higher scores. The main limitation of the deep learning context-aware recommender system is that it is unable to logically explain why the recommendation is made only when the result is delivered. Therefore, we plan to study how to reflect the reliability measured based on errors between the data restored with AE and the actual input data in the proposed model.

Author Contributions: Conceptualization, S.-Y.J. and Y.-K.K.; methodology, S.-Y.J.; investigation, S.-Y.J.; validation S.-Y.J. and Y.-K.K.; writing—review and editing, S.-Y.J.; supervision Y.-K.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by research fund of Chungnam National University.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://github.com/irecsys/CARSKit (accessed on 15 July 2020).

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

References

1. 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. [CrossRef]
2. Esmaeili, L.; Mardani, S.; Golpayegani, S.A.H.; Madar, Z.Z. A novel tourism recommender system in the context of social commerce. *Expert Syst. Appl.* **2020**, *149*, 113301. [CrossRef]
3. Jiang, P.; Zhu, Y.; Zhang, Y.; Yuan, Q. Life-stage prediction for product recommendation in e-commerce. In Proceedings of the Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 1879–1888. [CrossRef]
4. Adomavicius, G.; Tuzhilin, A. Context-aware recommender systems. In *Recommender Systems Handbook*; Springer: Boston, MA, USA, 2011; pp. 217–253.
5. Li, J.; Sun, C.; Lv, J. TCMF: Trust-Based Context-Aware Matrix Factorization for Collaborative Filtering. In Proceedings of the 2014 IEEE 26th International Conference on Tools with Artificial Intelligence, Limassol, Cyprus, 10–12 November 2014; pp. 815–821. [CrossRef]
6. Verbert, K.; Manouselis, N.; Ochoa, X.; Wolpers, M.; Drachsler, H.; Bosnic, I.; Duval, E. Context-aware recommender systems for learning: A survey and future challenges. *IEEE Trans. Learn. Technol.* **2012**, *5*, 318–335. [CrossRef]
7. Zheng, Y.; Mobasher, B.; Burke, R.D. Incorporating Context Correlation into Context-aware Matrix Factorization. In Proceedings of the 2015 International Conference on Constraints and Preferences for Configuration and Recommendation and Intelligent Techniques for Web Personalization, Buenos Aires, Argentina, 25–27 July 2015; Volume 1440.
8. Akhmatnurov, M.; Ignatov, D.I. Context-Aware Recommender System Based on Boolean Matrix Factorization. *CEUR Workshop Proc.* **2015**, *1466*, 99–110.
9. Hu, Y.; Koren, Y.; Volinsky, C. Collaborative filtering for implicit feedback datasets. In Proceedings of the 2008 Eighth IEEE International Conference on Data Mining, Pisa, Italy, 15–19 December 2008; pp. 263–272. [CrossRef]
10. Liu, Q.; Wu, S.; Wang, L. Cot: Contextual operating tensor for context-aware recommender systems. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; pp. 203–209.
11. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
12. Bokde, D.; Girase, S.; Mukhopadhyay, D. Matrix factorization model in collaborative filtering algorithms: A survey. *Procedia Comput. Sci.* **2015**, *49*, 136–146. [CrossRef]
13. Xue, H.J.; Dai, X.; Zhang, J.; Huang, S.; Chen, J. Deep Matrix Factorization Models for Recommender Systems. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, Melbourne, Australia, 19–25 August 2017; Volume 17, pp. 3203–3209. [CrossRef]
14. Mu, R.; Zeng, X.; Han, L. A survey of recommender systems based on deep learning. *IEEE Access* **2018**, *6*, 69009–69022. [CrossRef]
15. Tay, Y.; Luu, A.T.; Hui, S.C. Multi-Pointer Co-Attention Networks for Recommendation. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2309–2318. [CrossRef]
16. Frank, M.; Drikakis, D.; Charissis, V. Machine-learning methods for computational science and engineering. *Computation* **2020**, *8*, 15. [CrossRef]
17. Zhang, S.; Yao, L.; Sun, A.; Tay, Y. Deep learning based recommender system: A survey and new perspectives. *ACM Comput. Surv. (CSUR)* **2019**, *52*, 1–38. [CrossRef]
18. Kim, D.; Park, C.; Oh, J.; Lee, S.; Yu, H. Convolutional matrix factorization for document context-aware recommendation. In Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, 15–19 September 2016; pp. 233–240. [CrossRef]
19. Bobadilla, J.; Alonso, S.; Hernando, A. Deep Learning Architecture for Collaborative Filtering Recommender Systems. *Appl. Sci.* **2020**, *10*, 2441. [CrossRef]
20. Shoja, B.M.; Tabrizi, N. Customer reviews analysis with deep neural networks for e-commerce recommender systems. *IEEE Access* **2019**, *7*, 119121–119130. [CrossRef]
21. Tuinhof, H.; Pirker, C.; Haltmeier, M. Image-based fashion product recommendation with deep learning. In Proceedings of the International Conference on Machine Learning, Optimization, and Data Science, Volterra, Italy, 13–16 September 2018; Springer: Cham, Swizerland, 2018; pp. 472–481. [CrossRef]
22. Wang, H.; Wang, N.; Yeung, D.Y. Collaborative deep learning for recommender systems. In Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, 10–13 August 2015; pp. 1235–1244.
23. Baltrunas, L.; Ludwig, B.; Ricci, F. Matrix factorization techniques for context aware recommendation. In Proceedings of the Fifth ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 301–304. [CrossRef]
24. Rendle, S.; Schmidt-Thieme, L. Pairwise interaction tensor factorization for personalized tag recommendation. In Proceedings of the Third ACM International Conference on Web Search and Data Mining, New York, NY, USA, 3–6 February 2010; pp. 81–90. [CrossRef]
25. Karatzoglou, A.; Amatriain, X.; Baltrunas, L.; Oliver, N. Multiverse recommendation: N-dimensional tensor factorization for context-aware collaborative filtering. In Proceedings of the Fourth ACM Conference on Recommender Systems, Barcelona, Spain, 26–30 September 2010; pp. 79–86. [CrossRef]
26. Baltrunas, L.; Ricci, F. Experimental evaluation of context-dependent collaborative filtering using item splitting. *User Modeling User-Adapt. Interact.* **2014**, *24*, 7–34. [CrossRef]

27. Phuong, T.M.; Phuong, N.D. Graph-based context-aware collaborative filtering. *Expert Syst. Appl.* **2019**, *126*, 9–19. [CrossRef]
28. Rendle, S.; Gantner, Z.; Freudenthaler, C.; Schmidt-Thieme, L. Fast context-aware recommendations with factorization machines. In Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, Beijing, China, 24–28 July 2011; pp. 635–644. [CrossRef]
29. He, X.; Liao, L.; Zhang, H.; Nie, L.; Hu, X.; Chua, T.S. Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web, Perth, Australia, 3–7 April 2017; pp. 173–182.
30. 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. [CrossRef]
31. Zhong, S.-T.; Huang, L.; Wang, C.-D.; Lai, J.-H.; Yu, P.S. An autoencoder framework with attention mechanism for cross-domain recommendation. *IEEE Trans. Cybern.* **2020**, 1–13. [CrossRef] [PubMed]
32. Pan, Y.; He, F.; Yu, H. Learning social representations with deep autoencoder for recommender system. *World Wide Web* **2020**, *23*, 2259–2279. [CrossRef]
33. Zheng, Y.; Mobasher, B.; Burke, R. Carskit: A java-based context-aware recommendation engine. In Proceedings of the 2015 IEEE International Conference on Data Mining Workshop (ICDMW), Atlantic City, NJ, USA, 14–17 November 2015; pp. 1668–1671. [CrossRef]