electronics

MDPI

Article

Explainable Neural Tensor Factorization for Commercial Alley Revenues Prediction

Minkyu Kim [1], Suan Lee [2,*] and Jinho Kim [3]

[1] Clinical Decision Support Systems Team, Ziovision Inc., Chuncheon 24341, Republic of Korea; minkyu.kim@ziovision.co.kr
[2] School of Computer Science, Semyung University, Jecheon 27136, Republic of Korea
[3] Department of Computer Science and Engineering, Kangwon National University, Chuncheon 24341, Republic of Korea; jhkim@kangwon.ac.kr
* Correspondence: suanlee@semyung.ac.kr

Abstract: Many individuals aspire to start their own businesses and achieve financial success. Before launching a business, they must decide on a location and the type of service to offer. This decision requires collecting and analyzing various characteristics of potential locations and services, such as average revenues and foot traffic. However, this process is challenging because it demands expert knowledge in data collection and analysis. To address this issue, we propose Neural Tensor Factorization (NeuralTF) and Explainable Neural Tensor Factorization (XNeuralTF). These methods automatically analyze these characteristics and predict revenues. NeuralTF integrates Tensor Factorization (TF) with Multi-Layer Perceptron (MLP). This integration allows it to handle multi-dimensional tensors effectively. It also learns both explicit and implicit higher-order feature interactions, leading to superior predictive performance. XNeuralTF extends NeuralTF by providing explainable recommendations for three-dimensional tensors. Additionally, we introduce two novel metrics to evaluate the explainability of recommendation models. We conducted extensive experiments to assess both predictive performance and explainability. Our results show that XNeuralTF achieves comparable or superior performance to state-of-the-art methods, while also offering the highest level of explainability.

Keywords: explainable AI; tensor factorization; neural networks; deep learning; recommender systems

check for updates

Citation: Kim, M.; Lee, S.; Kim, J. Explainable Neural Tensor Factorization for Commercial Alley Revenues Prediction. *Electronics* **2024**, *13*, 3279. https://doi.org/10.3390/electronics13163279

Academic Editor: Maciej Ławryńczuk

Received: 8 July 2024
Revised: 10 August 2024
Accepted: 14 August 2024
Published: 19 August 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Many aspire to start their own businesses and achieve financial success. To succeed, it is essential to collect and analyze the characteristics of various locations and services. This analysis helps determine the most appropriate location and the desired type of service. Factors such as average revenues, foot traffic, and the services offered by surrounding stores are crucial in making these decisions. However, collecting and analyzing these data is notoriously challenging for individuals. This task requires considerable time, effort, and expert knowledge.

An automated system that analyzes the characteristics of locations and services would be immensely beneficial. Such a system could recommend appropriate choices, helping prospective business owners and boosting the local economy. Furthermore, it is important for users to understand the reasoning behind the system's recommendations. Hence, a system that can explain its recommendation results is also required.

Recent advancements in machine learning have introduced methods capable of automatically analyzing data characteristics [1–3]. These methods reduce the user's effort in data analysis and feature engineering. They do this by learning multiple feature interactions simultaneously, offering remarkable performance. However, these methods are often unsuitable for handling multi-dimensional tensors. They can only learn limited orders of

feature interactions. Additionally, they lack the capability to explain why certain items were recommended.

To efficiently analyze complex real-world data and provide more accurate recommendations, a new method is needed. This method should handle multi-dimensional tensors and be capable of learning higher-order feature interactions. Moreover, an explainable recommender system is necessary. Such a system can elucidate why specific items were recommended, thereby enhancing the system's reliability.

In this paper, we propose two novel machine learning methods: Neural Tensor Factorization (NeuralTF) and Explainable Neural Tensor Factorization (XNeuralTF). NeuralTF combines Tensor Factorization (TF) with a Multi-Layer Perceptron (MLP). This combination makes it suitable for handling multi-dimensional tensors. It also simultaneously learns explicit and implicit higher-order feature interactions. XNeuralTF extends NeuralTF to provide explanations for recommendation results in a three-dimensional tensor. It offers neighbor-based explanations by referencing neighboring items of a recommended item.

In addition, we introduce two novel metrics to evaluate the explainability of recommender systems. We conducted extensive experiments to compare the performance and explainability of various recommender systems using the Seoul Commercial Alley dataset [4].

In summary, the major contributions of this paper are as follows:

- We propose NeuralTF, which is suitable for handling multi-dimensional tensors and simultaneously learns explicit and implicit higher-order feature interactions.
- We introduce XNeuralTF, an extension of NeuralTF that explains recommendation results for a three-dimensional tensor.
- We suggest two novel metrics to evaluate the explainability of recommender systems.
- We conduct experiments using a real-world dataset to compare the performance and explainability of various recommender systems.

2. Related Works

2.1. Factorization Methods

Matrix factorization (MF) [5–7] decomposes a matrix into several smaller matrices. These smaller matrices are then multiplied to approximate and reconstruct the original matrix. Traditionally, MF methods have been widely used in various fields, such as computer vision and recommender systems. SVD++ [8] is an improved form of MF that incorporates implicit feedback. TimeSVD++ [9] further extends SVD++ to account for temporal dynamics. Both SVD++ and timeSVD++ have achieved remarkable success in collaborative filtering. A top-n recommender system using matrix factorization (also known as matrix completion) was proposed by [10]. However, one major disadvantage of matrix factorization is its limitation to two-dimensional data.

Tensor Factorization (TF) [11,12] extends this concept to tensors. It decomposes a tensor into several smaller tensors, with learning and inference processes that are very similar to those of MF. Tucker Decomposition [13,14] extends SVD to handle multi-dimensional tensors. A study on estimating the revenues of Seoul commercial alleys using Tucker Decomposition was conducted in [15]. Factorization Machine (FM) [3,16] decomposes a multi-dimensional tensor into latent vectors corresponding to each dimension. Unlike most factorization methods, FM can handle continuous features. FM has shown better performance in recommendation tasks. Extensive experiments and comparisons of various factorization methods were conducted in [17]. Despite the recent dominance of deep learning, traditional factorization methods continue to inspire researchers.

2.2. Deep Learning for Recommender Systems

Recently, deep learning-based recommender systems have been widely studied. These systems have been explored in fields such as collaborative filtering and click-through-rate (CTR) prediction. They often outperform traditional methods. For example, a collaborative

filtering system using an auto-encoder was proposed in [18]. Additionally, deep learning-based matrix factorization was introduced in [19].

DeepCrossing [20] is a deep learning-based recommender system that utilizes residual connections. It has shown good performance in CTR prediction without relying on hand-crafted features. Furthermore, a session-based recommender system that considers user history information was proposed in [21]. A comprehensive survey of deep learning-based recommender systems was conducted in [22].

Despite these advancements, most deep learning-based and traditional recommender systems still require hand-crafted features to achieve optimal performance.

2.3. Combinatorial Methods

Various methods to automatically learn feature interactions have been proposed. These methods can be seen as combinations of several models that learn different feature interactions. FM [3,16] can be viewed as a combination of linear regression and second-order factorization methods. Wide & Deep Learning [23] combines linear regression and MLP. This combination allows the model to simultaneously learn first-order and higher-order feature interactions. An improved version of Wide & Deep Learning, which shares the embedding space and shows good performance in regression analysis, was proposed in [24].

Neural Collaborative Filtering (NCF) [2] combines matrix factorization and MLP. This approach demonstrates impressive performance in collaborative filtering. Cross Network [25] has a multilayer architecture but remains a linear model. It explicitly learns higher-order feature interactions. Neural Factorization Machine (NFM) [26] extends FM by using a deep neural network and bi-interaction pooling. DeepFM [1] combines FM and MLP.

Compressed Interaction Network (CIN) [27] is similar to Cross Network, but it makes interactions in a vector-wise manner. xDeepFM [27] combines CIN and MLP. Our previous work [28] proposed NeuralTF, which combines TF and MLP. We conducted experiments on the Seoul Commercial Alley dataset.

The Adaptive Factorization Network (AFN) [29] is a model that automatically learns and weighs arbitrary-order feature interactions using a novel logarithmic transformation layer, resulting in improved predictive performance across various datasets.

EulerNet [30] is a model that leverages Euler's formula to adaptively learn arbitrary-order feature interactions in a complex vector space, enabling more efficient and accurate predictions in tasks like click-through rate (CTR) estimation.

Combinatorial methods significantly outperform existing recommender systems. However, they are still unsuitable for handling multi-dimensional tensors. These methods can only learn bounded orders of feature interactions. Additionally, they cannot explain why certain items were recommended.

2.4. Explainable Methods for Recommender Systems

Users may not trust recommendation results from unexplainable recommender systems. To address this issue, various explainable recommendation methods have been proposed. Explainable matrix factorization (EMF) [31,32] constrains MF so that embedding vectors of similar users (or items) are close to each other. This approach provides neighbor-based explanations.

Tree-enhanced Embedding Model (TEM) [33] utilizes decision trees and attentive deep neural networks to create an explainable recommender system. FacT [34] is a latent factor model that learns explanatory rules using a regression tree. An explainable matrix factorization method for recommending novel items was proposed in [35]. A survey of explainable recommender systems was conducted in [36].

3. Proposed Method

In this section, we introduce two novel models: Neural Tensor Factorization (NeuralTF) and Explainable Neural Tensor Factorization (XNeuralTF). NeuralTF combines Tensor Factorization (TF) and Multi-Layer Perceptron (MLP) components. Both TF and MLP are

highly suitable for handling multi-dimensional tensors. TF can explicitly learn higher-order feature interactions. Meanwhile, MLP can implicitly learn higher-order feature interactions. By leveraging both TF and MLP, NeuralTF effectively handles multi-dimensional tensors and achieves high performance.

XNeuralTF extends NeuralTF to explain recommendation results for a 3-dimensional tensor. It ensures that similar items are embedded close to each other. Additionally, it provides neighbor-based explanations.

3.1. Embedding Layer

Categorical features must be encoded as real vectors before being fed into a machine learning model. The simplest method is one-hot encoding. However, one-hot vectors are sparse and high-dimensional, which can lead to issues such as slower computational speed and increased memory usage. To address these issues, an embedding method is used. This method encodes categorical features into low-dimensional dense vectors.

Our embedding layer is defined as follows:

$$e_i = W_i^T x_i, \tag{1}$$

where x_i is the one-hot vector of the i-th feature, W_i is the embedding weights of the i-th feature, and e_i is the embedding vector of the i-th feature. In our method, an embedding vector is the weighted sum of a one-hot vector. Figure 1 shows the visualization of our embedding layer.

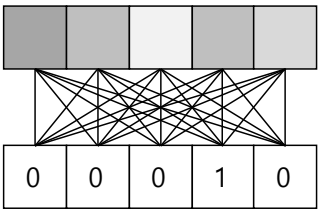


Figure 1. The embedding layer.

3.2. Tensor Factorization Component

Tensor Factorization (TF) decomposes a tensor into several smaller tensors. There are various TF methods, including Tucker Decomposition [13,14] and Non-negative TF [11]. Our TF method decomposes a tensor into vectors corresponding to each dimension. Each vector contains information about its corresponding dimension.

Our TF component is defined as follows:

$$f_{tf}(\mathbf{e}) = \mathbf{e}_1 \odot \mathbf{e}_2 \odot \ldots \odot \mathbf{e}_N, \tag{2}$$

where $\mathbf{e}$ is the set of embedding vectors, N is the number of features, and $\odot$ is the Hadamard product. TF is well suited to handling multi-dimensional tensors. It can explicitly learn higher-order feature interactions due to the absence of non-linear functions in the TF component.

3.3. Multi-Layer Perceptron Component

The Multi-Layer Perceptron (MLP) is a deep learning method widely used for handling tabular data. Its multi-layer structure and non-linear transformations through activation functions give it high capacity. This allows MLP to implicitly learn higher-order feature interactions.

Our MLP component is defined as follows:

$$h_i(x) = \sigma(W_i^T x + b_i), \tag{3}$$

$$f_{mlp}(\mathbf{e}, \mathbf{z}) = h_L(\ldots h_1([\mathbf{e}_1; \ldots; \mathbf{e}_N; \mathbf{z}_1; \ldots; \mathbf{z}_M])), \tag{4}$$

where **e** is the set of embedding vectors and **z** is the set of extra features. h_i represents the i-th hidden layer, W_i are the trainable weights of the i-th layer, and b_i is the trainable bias of the i-th layer. σ denotes the activation function. N is the number of features, M is the number of extra features, L is the layer depth, and $[\cdot;\cdot]$ is the concatenation operator.

3.4. Neural Tensor Factorization

NeuralTF combines the TF and MLP components, taking advantage of both. NeuralTF is defined as follows:

$$f_{neuraltf}(\mathbf{e}, \mathbf{z}) = W^T[f_{tf}(\mathbf{e}); f_{mlp}(\mathbf{e}, \mathbf{z})] + b, \tag{5}$$

where f_{tf} is the TF component and f_{mlp} is the MLP component. W represents the trainable output weights and b is the trainable output bias.

Note that extra features are only fed into the MLP component. The two components are combined linearly. Figure 2 shows the architecture of NeuralTF.

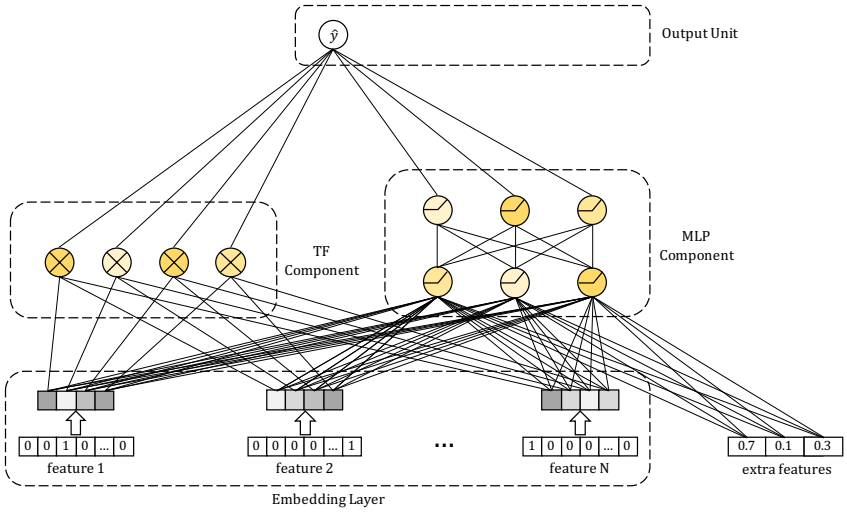


Figure 2. The architecture of NeuralTF.

3.5. Explainable Neural Tensor Factorization

XNeuralTF provides explanations for recommendation results for a 3-dimensional tensor. In our case, each dimension of the tensor represents *alley name*, *service name*, and *date*. We introduce two explanation methods: alley-based explanations and service-based explanations.

The alley-based explanation for commercial alley recommendations is defined as follows:

$$Pr(R_{\alpha,s,t} \geq \bar{R}_{a,:,t} \mid \alpha \in N_a) = \frac{|R_{\alpha,s,t} \geq \bar{R}_{a,:,t}|}{|N_a|}, \tag{6}$$

$$Expl_{a,s,t} = \begin{cases} Pr(R_{\alpha,s,t} \geq \bar{R}_{a,:,t} \mid \alpha \in N_a) & Pr > \theta \\ 0 & \text{otherwise} \end{cases}, \tag{7}$$

where $R_{a,s,t}$ is the revenue of service s in alley a on date t. $\bar{R}_{a,:,t}$ is the average revenue of alley a on date t for all services. N_a is the set of neighboring alleys of a. Lastly, θ is the threshold that determines whether a is explainable.

Similarly, the service-based explanation for service recommendation is defined as follows:

$$Pr(R_{a,\varsigma,t} \geq \bar{R}_{:,s,t} \mid \varsigma \in N_s) = \frac{|R_{a,\varsigma,t} \geq \bar{R}_{:,s,t}|}{|N_s|}, \tag{8}$$

$$Expl_{a,s,t} = \begin{cases} Pr(R_{a,\varsigma,t} \geq \bar{R}_{:,s,t} \mid \varsigma \in N_s) & Pr > \theta \\ 0 & \text{otherwise} \end{cases}, \tag{9}$$

where $\bar{R}_{:,s,t}$ is the average revenue of service s on date t for all alleys. N_s is the set of neighboring services of s.

We assume that people who want to start their business aim to earn higher revenues than the average. Therefore, if an alley or service has neighbors with higher revenues than the average, it has high explainability. For example, if the number of neighbors $|N_a|$ is 10 and the number of neighbors with higher revenues than the average is 6, then the explainability of a is 0.6. Finally, we apply the threshold θ to determine which alleys or services are explainable.

To find the neighbors of a particular alley or service, we compute the similarities between embedding vectors using cosine similarity. Cosine similarity is defined as follows:

$$sim(e_a, e_{\alpha \in I}) = \frac{e_a \cdot e_\alpha}{||e_a|| \times ||e_\alpha||}, \tag{10}$$

where e_a is the embedding vector of an alley or service and I is the set of all alleys or services. While we use cosine similarity in this work, other similarity functions such as Euclidean distance can also be effective.

Existing TF and MLP methods do not ensure that neighbors are closely embedded. Therefore, we add the following constraint to the loss function:

$$\mathcal{J} = \frac{1}{|D|} \sum_{a,s,t \in D} (R_{a,s,t} - \hat{R}_{a,s,t})^2 + \lambda_1 ||\Theta||^2 + \lambda_2 (e_a - e_s)^2 Expl_{a,s,t}, \tag{11}$$

where D indicates the dataset, Θ is the set of trainable parameters, λ_1 is the regularization strength, λ_2 is the explainability strength, and $Expl_{a,s,t}$ indicates the explainability of service s in alley a on date t.

In Equation (11), the first term, $\frac{1}{|D|} \sum_{a,s,t \in D} (R_{a,s,t} - \hat{R}_{a,s,t})^2$, represents the mean squared error. The second term, $\lambda_1 ||\Theta||^2$, represents the squared-*L2* regularization. The third term, $\lambda_2 (e_a - e_s)^2 Expl_{a,s,t}$, represents the explainability constraint.

This constraint ensures that the embedding vectors of alley a and service s are embedded as closely as the explainability strength λ_2 dictates. In other words, alley a is embedded closely with other alleys that co-occur with service s. Similarly, service s is embedded closely with other services that co-occur with alley a.

Figure 3 shows the architecture of XNeuralTF. Algorithm 1 provides the high-level pseudocode for alley-based XNeuralTF.

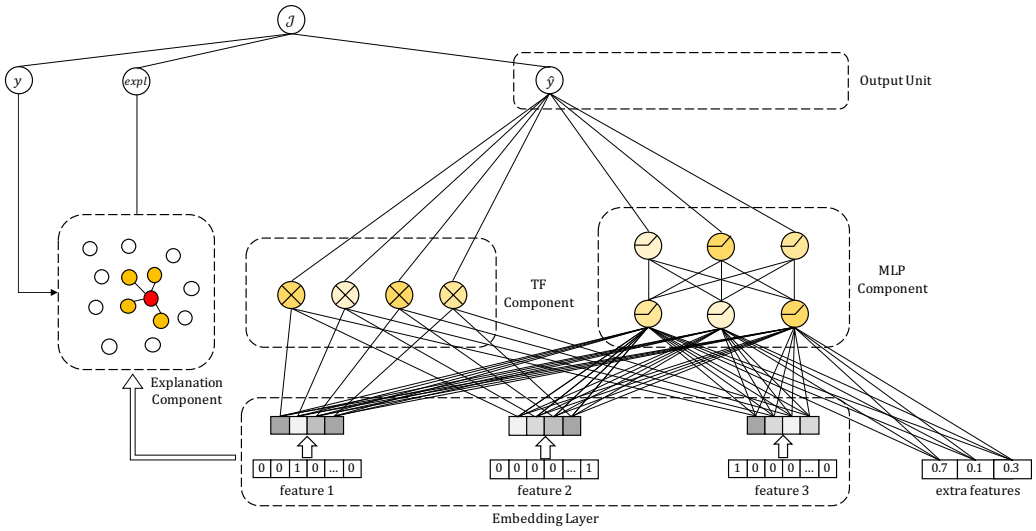


Figure 3. The architecture of XNeuralTF.

Algorithm 1 Pseudocode of alley-based XNeuralTF

Require: alley a, service s, date t, extra features z, threshold θ, and trainable parameters Θ

$e_a \leftarrow get\ the\ embedding\ vector\ of\ a$
$e_s \leftarrow get\ the\ embedding\ vector\ of\ s$
$e_t \leftarrow get\ the\ embedding\ vector\ of\ t$
$\hat{y}_{tf} \leftarrow compute\ TF\ component\ with\ e_a,\ e_s\ and\ e_t$
$\hat{y}_{mlp} \leftarrow compute\ MLP\ component\ with\ e_a,\ e_s,\ e_t\ and\ z(optional)$
$\hat{y} \leftarrow compute\ NeuralTF\ through\ W^T[\hat{y}_{tf}; \hat{y}_{mlp}] + b$
$N_a \leftarrow get\ top\text{-}k\ neighbors\ of\ e_a\ via\ cosine\ similarity$
$N_a^* \leftarrow get\ neighbors\ with\ higher\ revenues\ than\ the\ average\ from\ N_a$
$pr_{a,s,t} \leftarrow compute\ explainability\ through\ |N_a^*|/|N_a|$
$expl_{a,s,t} \leftarrow applying\ threshold\ \theta\ to\ pr_{a,s,t}$
$\mathcal{J} \leftarrow compute\ loss\ with\ \hat{y},\ \Theta\ and\ expl_{a,s,t}$
$\Theta \leftarrow update\ trainable\ parameters\ via\ \Theta - \eta \frac{\partial \mathcal{J}}{\partial \Theta}$

4. Experiments

4.1. Dataset

We use the Seoul Commercial Alley dataset [4] for our experiments. The dataset contains real-world information about a large number of alleys and stores in Seoul. It includes 1,573,026 data points and 111 features.

Unfortunately, the dataset has many missing values and noisy features. Therefore, we preprocess the dataset as follows:

- We drop duplicate features.
- We eliminate derived features that can be gathered by combining multiple features.
- We remove features directly related to revenues.
- We remove features that include missing values.
- We eliminate data points with revenues below or equal to zero.

After preprocessing, 116,193 data points and seven features remain. The remaining features are {*alley name*, *service name*, *date*, *overcrowding value*, *growth value*, *activity value*, and *stability value*}.

Overcrowding value indicates the risk of a new store. Activity value reflects the amount of transaction activity for a particular service. Growth value shows the growth rate of revenues for a particular commercial alley. Stability value indicates the certainty of survival for a store.

We construct a three-dimensional tensor using *alley name*, *service name*, and *date*. We use *overcrowding value*, *activity value*, *growth value*, and *stability value* as extra features. Figure 4 shows the map of Seoul and examples of data points.

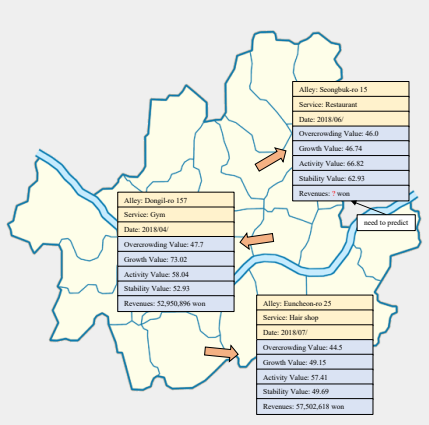


Figure 4. The map of Seoul and examples of data points.

4.2. Experimental Setup

We compare various machine learning models on the Seoul Commercial Alley dataset. We evaluate the models using a seven-fold cross-test. During the training period, we use 15% of the training data as validation data for hyper-parameter tuning.

Hyper-parameter tuning is essential for achieving high performance in machine learning models. However, it is notoriously difficult and time consuming. Many researchers spend a significant amount of time and effort on hyper-parameter tuning. Fortunately, a hyper-parameter optimization tool called Tune [37] is available.

We use Tune to search for the optimal hyper-parameter combination for each model, running 100 trials. Each model is trained for 1000 epochs. The Adam optimizer is used to optimize the trainable parameters, although other gradient-based optimizers also work well.

4.3. Performance Comparison

We compare the performances of various machine learning models with NeuralTF and XNeuralTF. The performances are measured using RMSE and MAE, which are defined as follows:

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

$$MAE = \frac{1}{|D|}\sum_{i=1}^{|D|}|y_i - \hat{y}_i|, \tag{13}$$

where y is the target value and $\hat{y}$ is the predicted value.

Table 1 shows the prediction performance of the different models, where the lowest value is shown in bold and the second lowest value is shown in underline. DeepFM shows the lowest RMSE, while service-based XNeuralTF achieves the lowest MAE with a lower standard deviation. The XNeuralTF(service) model achieves the second lowest RMSE, while the XNeuralTF(alley) model achieves the second lowest MAE.

The experimental results indicate that XNeuralTF matches or achieves a state-of-the-art result. Additionally, it is more stable and robust compared to other models.

Table 1. Performance comparison of different models.

	RMSE	**MAE**
Linear	0.764038 (±0.039668)	0.351001 (±0.009508)
TF	0.220472 (±0.046086)	0.093706 (±0.002539)
FM	0.269507 (±0.045068)	0.115827 (±0.002810)
MLP	0.205352 (±0.028200)	0.088146 (±0.002657)
Wide & Deep	0.205623 (±0.026090)	0.088506 (±0.002419)
DeepFM	**0.202959 (±0.028913)**	0.087444 (±0.002301)
AFN	0.360200 (±0.072595)	0.157997 (±0.019907)
EulerNet	0.207433 (±0.023316)	0.103130 (±0.004636)
NeuralTF	0.204925 (±0.027364)	0.087600 (±0.002782)
XNeuralTF (alley)	0.207087 (±0.028343)	<u>0.086300 (±0.002460)</u>
XNeuralTF (service)	<u>0.204597 (±0.027268)</u>	**0.085614 (±0.001445)**

In general, it is known that explainable models perform worse than unexplainable models. However, in our experiments, XNeuralTF outperforms other non-explainable models.

This improved performance is due to the explainability constraint. By constraining the embedding vectors of neighboring alleys and services to be close together, XNeuralTF can learn more meaningful embedding vectors.

4.4. Explainability Comparison

In this section, we compare the explainabilities of the experimental models. Since the linear model and the Wide component of Wide & Deep Learning cannot learn embedding vectors, they do not participate in this experiment.

Explainability Precision (EP) and Explainability Recall (ER) [31] are evaluation metrics for neighbor-based explanations. They are defined as follows:

$$EP = \frac{|I_{rec} \cap I_{exp}|}{|I_{rec}|}, \tag{14}$$

$$ER = \frac{|I_{rec} \cap I_{exp}|}{|I_{exp}|}, \tag{15}$$

where I_{rec} is the set of recommended items and I_{exp} is the set of explainable items.

However, EP and ER depend only on the number of explainable items. They cannot account for how strongly explainable each item is. Therefore, we suggest Weighted EP (WEP) and Weighted ER (WER) to consider the explainability of each item.

WEP and WER for alley-based explanations are defined as follows:

$$WEP = \frac{\sum_{\alpha \in I_{rec} \cap I_{exp}} Expl_{\alpha,s,t}}{|I_{rec}|}, \tag{16}$$

$$WER = \frac{\sum_{\alpha \in I_{rec} \cap I_{exp}} Expl_{\alpha,s,t}}{|I_{exp}|}, \tag{17}$$

where α indicates an explainable recommended alley.

WEP and WER for service-based explanations are defined as follows:

$$WEP = \frac{\sum_{\varsigma \in I_{rec} \cap I_{exp}} Expl_{a,\varsigma,t}}{|I_{rec}|}, \tag{18}$$

$$WER = \frac{\sum_{\varsigma \in I_{rec} \cap I_{exp}} Expl_{a,\varsigma,t}}{|I_{exp}|}, \tag{19}$$

where ς indicates an explainable recommended service.

Table 2 shows the explainabilities of different models, with the highest values in bold and the second highest in underline. NeuralTF shows the highest EP, while TF shows the highest ER in the experiment. However, XNeuralTF shows the highest WEP and WER. This indicates that XNeuralTF can more strongly vouch for the explainability of recommended items than other models.

Additionally, we gain an interesting insight from the experiment. Deep learning-based models provide higher explainability than non-deep learning models. This result shows that the embedding vectors learned by deep learning models are closely located to their neighbors' vectors. In other words, deep learning models learn higher quality embedding vectors than non-deep learning models.

Table 2. Explainability comparison of different models.

	WEP	**WER**
TF	0.423714	0.000220
FM	0.425143	0.000204
MLP	0.624571	0.000269
DeepFM	0.597714	0.000263
AFN	0.408286	0.000185
EulerNet	0.508857	0.000227
NeuralTF	<u>0.650000</u>	<u>0.000280</u>
XNeuralTF	**0.654286**	**0.000282**

4.5. Hyper-Parameter Study

In this section, we examine how the performance of XNeuralTF changes with variations in embedding size, the number of units, and the number of layers. These are the hyper-parameters that control the model's capacity. Additionally, we observe how changes in explainability strength, another important hyper-parameter of XNeuralTF, affect performance. In our experiments, we set the explainability strength λ_2 to 0.001. Based on the results of our hyper-parameter study, we recommend setting λ_2 to either 0.001 or 0.005 to achieve a balance between accuracy and explainability.

4.5.1. Embedding Size

The information of each feature is embedded in the corresponding embedding vector. If the embedding size is large, the embedding vector can contain more information. However, more data are required for training. On the other hand, if the embedding size is small, the embedding vector cannot contain enough information, leading to decreased performance.

Figure 5 shows the changes in validation loss, test RMSE, and test MAE for different embedding sizes. When the embedding size exceeds 128, there are no significant changes in test RMSE and test MAE, but validation loss increases. Therefore, the optimal embedding size in our case is 128.

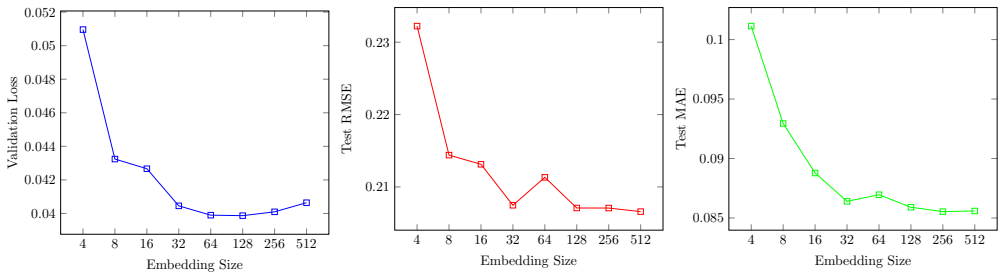


Figure 5. Changes in validation loss, test RMSE, and test MAE for different embedding sizes.

4.5.2. Number of Units

A small number of units in a layer can lead to a loss of information and reduce performance by learning poor representations. Conversely, overfitting may occur if the number of units in a layer is too large. Therefore, finding the appropriate number of units is crucial.

Figure 6 shows the changes in validation loss, test RMSE, and test MAE for different numbers of units. As the number of units increases, validation loss decreases. However, test RMSE increases when the number of units exceeds 512, and test MAE increases when the number of units exceeds 256. Thus, the optimal number of units in our case is 256.

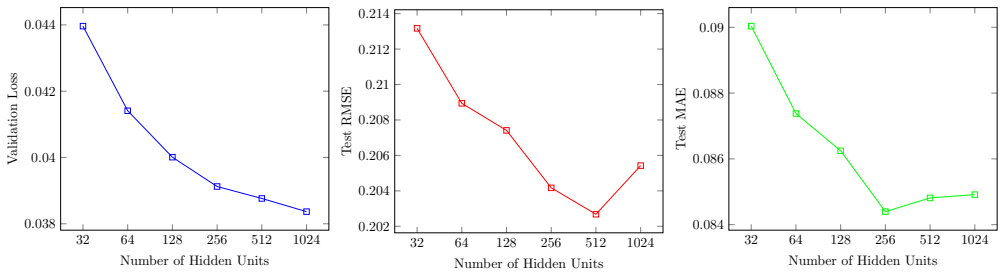


Figure 6. Changes in validation loss, test RMSE, and test MAE for different numbers of hidden units.

4.5.3. Number of Layers

The number of layers significantly influences the capacity of a machine learning model. Finding the appropriate number of layers is necessary to avoid both underfitting and overfitting.

Figure 7 shows the changes in validation loss, test RMSE, and test MAE for different numbers of layers. The validation loss and test RMSE show the best performance when the number of layers is 1. Performance generally improves as the number of layers increases, except when there is only 1 layer. Further study is required to understand why these results occurred.

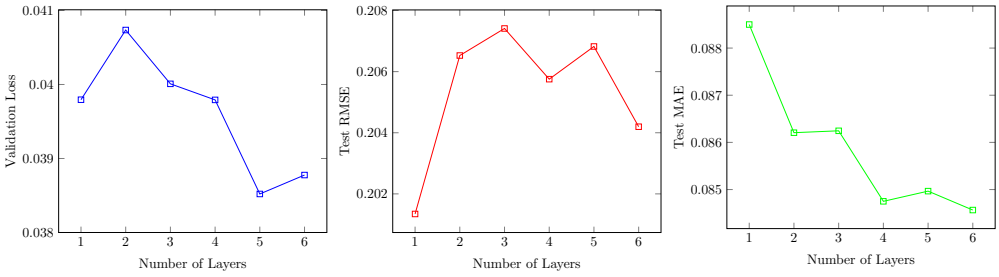


Figure 7. Changes in validation loss, test RMSE, and test MAE for different numbers of layers.

4.5.4. Explainability Strength

Explainability strength is a hyper-parameter that controls the explainability of XNeuralTF. A higher explainability strength indicates a stronger explainability constraint. Therefore, there is a trade-off between explainability strength and predictive performance.

Figure 8 shows the changes in validation loss, test RMSE, and test MAE for different explainability strengths. We can observe that predictive performance decreases as explainability strength increases.

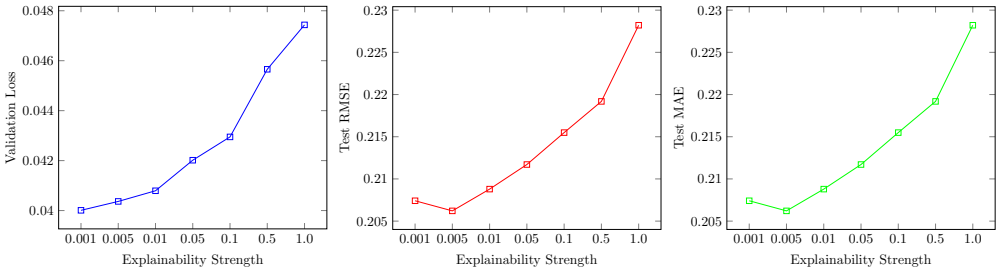


Figure 8. Changes in validation loss, test RMSE, and test MAE for different explainability strengths.

5. Conclusions

Collecting and analyzing the characteristics of locations and services are essential for a successful business. However, this process is notoriously difficult for individuals. To overcome these challenges, we proposed two novel recommender systems: NeuralTF and XNeuralTF.

NeuralTF is well suited to handling multi-dimensional tensors. It shows high predictive performance by simultaneously learning explicit and implicit higher-order feature interactions. XNeuralTF is an extended version of NeuralTF. It provides neighbor-based explanations for three-dimensional tensors.

We compared various machine learning models with NeuralTF and XNeuralTF. The experimental results demonstrated that XNeuralTF matched or achieved state-of-the-art results with the highest level of explainability.

In this paper, we conducted experiments on the Seoul Commercial Alley dataset. However, we expect that our XNeuralTF will also perform well in other recommendation tasks, such as movie or book recommendations.

Author Contributions: Conceptualization, S.L. and M.K.; methodology, S.L. and M.K.; software, M.K.; validation, S.L. and M.K.; formal analysis, M.K.; investigation, S.L.; resources, S.L.; data curation, S.L.; writing—original draft preparation, M.K.; writing—review and editing, S.L.; visualization, M.K.; supervision, J.K.; project administration, J.K.; funding acquisition, S.L. All authors have read and agreed with the published version of the manuscript.

Funding: This paper was supported by the Semyung University Research Grant of 2023.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon request from the corresponding author.

Conflicts of Interest: The authors declare no conflicts of interest. Author Minkyu Kim was employed by the company Ziovision Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Guo, H.; Tang, R.; Ye, Y.; Li, Z.; He, X. DeepFM: A Factorization-Machine based Neural Network for CTR Prediction. In Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17, Melbourne, Australia, 19–25 August 2017; pp. 1725–1731.
2. 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.
3. Rendle, S. Factorization Machines. In Proceedings of the 2010 IEEE International Conference on Data Mining, Sydney, Australia, 13–17 December 2010; pp. 995–1000.
4. Seoul Open Data Square. 2019. Available online: https://data.seoul.go.kr/ (accessed on 7 June 2022).
5. Koren, Y.; Bell, R.; Volinsky, C. Matrix Factorization Techniques for Recommender Systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
6. Lee, D.D.; Seung, H.S. Learning the Parts of Objects by Non-Negative Matrix Factorization. *Nature* **1999**, *401*, 788–791. [CrossRef] [PubMed]
7. Paatero, P.; Tapper, U. Positive Matrix Factorization: A Non-Negative Factor Model with Optimal Utilization of Error Estimates of Data Values. *Environmetrics* **1994**, *5*, 111–126. [CrossRef]
8. Koren, Y. Factorization Meets the Neighborhood: A Multifaceted Collaborative Filtering Model. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 426–434.
9. Koren, Y. Collaborative Filtering with Temporal Dynamics. In Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, 28 June–1 July 2009; pp. 447–456.
10. Kang, Z.; Peng, C.; Cheng, Q. Top-N Recommender System via Matrix Completion. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; pp. 179–184.
11. Shashua, A.; Hazan, T. Non-Negative Tensor Factorization with Applications to Statistics and Computer Vision. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 792–799.
12. Welling, M.; Weber, M. Positive Tensor Factorization. *Pattern Recognit. Lett.* **2001**, *22*, 1255–1261. [CrossRef]
13. De Lathauwer, L.; De Moor, B.; Vandewalle, J. A Multilinear Singular Value Decomposition. *SIAM J. Matrix Anal. Appl.* **2000**, *21*, 1253–1278. [CrossRef]
14. Kim, Y.D.; Choi, S. Nonnegative Tucker Decomposition. In Proceedings of the 2007 IEEE Conference on Computer Vision and Pattern Recognition, Minneapolis, MN, USA, 17–22 June 2007; pp. 1–8.
15. Park, S.; Lee, S.; Kim, J. Estimating Revenues of Seoul Commercial Alley Services using Tensor Decomposition & Generating Recommendation System. In Proceedings of the 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Republic of Korea, 19–22 February 2020; pp. 287–294.
16. Rendle, S. Factorization Machines with libFM. *ACM Trans. Intell. Syst. Technol. (TIST)* **2012**, *3*, 1–22. [CrossRef]
17. Rendle, S.; Zhang, L.; Koren, Y. On the Difficulty of Evaluating Baselines: A Study on Recommender Systems. *arXiv* **2019**, arXiv:1905.01395.
18. Wu, Y.; DuBois, C.; Zheng, A.X.; Ester, M. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, San Francisco, CA, USA, 22–25 February 2016; pp. 153–162.
19. 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, IJCAI-17, Melbourne, Australia, 19–25 August 2017; pp. 3203–3209.
20. Shan, Y.; Hoens, T.R.; Jiao, J.; Wang, H.; Yu, D.; Mao, J. Deep Crossing: Web-Scale Modeling without Manually Crafted Combinatorial Features. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 255–262.
21. Hidasi, B.; Karatzoglou, A.; Baltrunas, L.; Tikk, D. Session-Based Recommendations with Recurrent Neural Networks. *arXiv* **2015**, arXiv:1511.06939.
22. 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]

23. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & Deep Learning for Recommender Systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
24. Kim, M.; Lee, S.; Kim, J. A Wide & Deep Learning Sharing Input Data for Regression Analysis. In Proceedings of the 2020 IEEE International Conference on Big Data and Smart Computing (BigComp), Busan, Republic of Korea, 19–22 February 2020; pp. 8–12.
25. Wang, R.; Fu, B.; Fu, G.; Wang, M. Deep & Cross Network for Ad Click Predictions. In Proceedings of the ADKDD'17, Halifax, NS, Canada, 13–17 August 2017; pp. 1–7.
26. He, X.; Chua, T.S. Neural Factorization Machines for Sparse Predictive Analytics. In Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Tokyo, Japan, 7–11 August 2017; pp. 355–364.
27. Lian, J.; Zhou, X.; Zhang, F.; Chen, Z.; Xie, X.; Sun, G. xDeepFM: Combining Explicit and Implicit Feature Interactions for Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018.
28. Kim, M.; Lee, S. Predicting Revenues of Seoul Commercial Alley Using Neural Tensor Factorization. In Proceedings of the 2021 IEEE International Conference on Big Data and Smart Computing (BigComp), Jeju Island, Republic of Korea, 17–20 January 2021; pp. 192–195.
29. Cheng, W.; Shen, Y.; Huang, L. Adaptive factorization network: Learning adaptive-order feature interactions. In Proceedings of the AAAI Conference on Artificial Intelligence, Philadelphia, PA, USA, 27 February–2 March 2020; Volume 34, pp. 3609–3616.
30. Tian, Z.; Bai, T.; Zhao, W.X.; Wen, J.R.; Cao, Z. EulerNet: Adaptive Feature Interaction Learning via Euler's Formula for CTR Prediction. In Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval, Taipei, Taiwan, 23–27 July 2023; pp. 1376–1385.
31. Abdollahi, B.; Nasraoui, O. Explainable Matrix Factorization for Collaborative Filtering. In Proceedings of the 25th International Conference Companion on World Wide Web, Montreal, QC, Canada, 11–15 May 2016; pp. 5–6.
32. Abdollahi, B.; Nasraoui, O. Using Explainability for Constrained Matrix Factorization. In Proceedings of the Eleventh ACM Conference on Recommender Systems, Como, Italy, 27–31 August 2017; pp. 79–83.
33. Wang, X.; He, X.; Feng, F.; Nie, L.; Chua, T.S. Tem: Tree-enhanced Embedding Model for Explainable Recommendation. In Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018; pp. 1543–1552.
34. Tao, Y.; Jia, Y.; Wang, N.; Wang, H. The FacT: Taming Latent Factor Models for Explainability with Factorization Trees. In Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval, Paris, France, 21–25 July 2019; pp. 295–304.
35. Coba, L.; Symeonidis, P.; Zanker, M. Personalised Novel and Explainable Matrix Factorisation. *Data Knowl. Eng.* **2019**, *122*, 142–158.
36. Zhang, Y.; Chen, X. Explainable Recommendation: A Survey and New Perspectives. *Found. Trends Inf. Retr.* **2020**, *14*, 1–101. [CrossRef]
37. Liaw, R.; Liang, E.; Nishihara, R.; Moritz, P.; Gonzalez, J.E.; Stoica, I. Tune: A Research Platform for Distributed Model Selection and Training. *arXiv* **2018**, arXiv:1807.05118.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.