TEM: Tree-enhanced Embedding Model for Explainable Recommendation

Xiang Wang
National University of Singapore
xiangwang@u.nus.edu

Xiangnan He*
National University of Singapore
xiangnanhe@gmail.com

Fuli Feng
National University of Singapore
fulifeng93@gmail.com

Liqiang Nie
ShanDong University
nieliqiang@gmail.com

Tat-Seng Chua
National University of Singapore
dcscts@nus.edu.sg

ABSTRACT

While collaborative filtering is the dominant technique in personalized recommendation, it models user-item interactions only and cannot provide concrete reasons for a recommendation. Meanwhile, the rich side information affiliated with user-item interactions (*e.g.,* user demographics and item attributes), which provide valuable evidence that why a recommendation is suitable for a user, has not been fully explored in providing explanations.

On the technical side, embedding-based methods, such as Wide&Deep and neural factorization machines, provide state-of-the-art recommendation performance. However, they work like a black-box, for which the reasons underlying a prediction cannot be explicitly presented. On the other hand, tree-based methods like decision trees predict by inferring decision rules from data. While being explainable, they cannot generalize to unseen feature interactions thus fail in collaborative filtering applications.

In this work, we propose a novel solution named *Tree-enhanced Embedding Method* that combines the strengths of embedding-based and tree-based models. We first employ a tree-based model to learn explicit decision rules (*aka.* cross features) from the rich side information. We next design an embedding model that can incorporate explicit cross features and generalize to unseen cross features on user ID and item ID. At the core of our embedding method is an easy-to-interpret attention network, making the recommendation process fully transparent and explainable. We conduct experiments on two datasets of tourist attraction and restaurant recommendation, demonstrating the superior performance and explainability of our solution.

CCS CONCEPTS

• **Information systems → Recommender systems**;

KEYWORDS

Explainable Recommendation, Tree-based Model, Embedding-based Model, Neural Attention Network

*Xiangnan He is the corresponding author.

WWW 2018, April 23–27, 2018, Lyon, France

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186066

ACM Reference Format:
Xiang Wang, Xiangnan He, Fuli Feng, Liqiang Nie, and Tat-Seng Chua. 2018. TEM: Tree-enhanced Embedding Model for Explainable Recommendation. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186066

1 INTRODUCTION

Personalized recommendation is at the core of many online customer-oriented services, such as E-commerce, social media, and content-sharing websites. Technically speaking, the recommendation problem is usually tackled as a matching problem, which aims to estimate the relevance score between a user and an item based on their available profiles. Regardless of the application domain, a user's profile usually consists of an ID (to identify which specific user) and some side information like age, gender, and income level. Similarly, an item's profile typically contains an ID and some attributes like category, tags, and price.

Collaborative filtering (CF) is the most prevalent technique for building a personalized recommendation system [21, 26]. It leverages users' interaction histories on items to select the relevant items for a user. From the matching view, CF uses the ID information only as the profile for a user and an item, and forgoes other side information. As such, CF can serve as a generic solution for recommendation without requiring any domain knowledge. However, the downside is that it lacks necessary reasoning or explanations for a recommendation. Specially, the explanation mechanisms are either *because your friend also likes it* (*i.e.,* user-based CF [24]) or *because the item is similar to what you liked before* (*i.e.,* item-based CF [35]), which are too coarse-grained and may be insufficient to convince users on a recommendation [14, 39, 45].

To persuade users to perform actions on a recommendation, we believe it is crucial to provide more concrete reasons in addition to similar users or items. For example, we recommend *iPhone 7 Rose Gold* to user *Emine*, because we find females aged 20-25 with a monthly income over $10,000 (which are *Emine*' demographics) generally prefer Apple products of pink color. To supercharge a recommender system with such informative reasons, the underlying recommender shall be able to (i) **explicitly** discover effective cross features from the rich side information of users and items, and (ii) estimate user-item matching score in an **explainable** way. In addition, we expect the use of side information will help in improving the performance of recommendation.

Nevertheless, none of existing recommendation methods can satisfy the above two conditions together. In the literature,

embedding-based methods such as matrix factorization [23, 26, 34] is the most popular CF approach, owing to the strong power of embeddings in generalizing from sparse user-item relations. Many variants have been proposed to incorporate side information, such as factorization machine (FM) [32], Neural FM [20], Wide&Deep [12], and Deep Crossing [36]. While these methods can learn feature interactions from raw data, we argue that the cross feature effects are only captured in a rather implicit way during the learning process; and most importantly, the cross features cannot be explicitly presented [36]. Moreover, existing works on using side information have mainly focused on the cold-start issue [5], leaving the explanation of recommendation relatively less touched.

In this work, we aim to fill the research gap by developing a recommendation solution that is both accurate and explainable. By **accurate**, we expect our method to achieve the same level of performance as existing embedding-based approaches [32, 36]. By **explainable**, we would like our method to be transparent in generating a recommendation and is capable of identifying the key cross features for a prediction. Towards this end, we propose a novel solution named *Tree-enhanced Embedding Method* (TEM), which combines embedding-based methods with decision tree-based approaches. First, we build a *gradient boosting decision trees* (GBDT) on the side information of users and items to derive effective cross features. We then feed the cross features into an embedding-based model, which is a carefully designed neural attention network that reweights the cross features according to the current prediction. Owing to the explicit cross features extracted by GBDTs and the easy-to-interpret attention network, the overall prediction process is fully transparent and self-explainable. Particularly, to generate reasons for a recommendation, we just need to select the most predictive cross features based on their attention scores.

As a main technical contribution, this work presents a new scheme that unifies the strengths of embedding-based and tree-based methods for recommendation. Embedding-based methods are known to have strong generalization ability [12, 20], especially in predicting the unseen crosses on user ID and item ID (*i.e.,* capturing the CF effect). However, when operating on the rich side information, embedding-based methods lose the important property of explainability — the cross features that contribute most to the prediction cannot be revealed. On the other hand, tree-based methods predict by generating explicit decision rules, making the resultant cross features directly interpretable. While such a way is highly suitable for learning from side information, it fails to predict unseen cross features, thus being unsuitable for incorporating user ID and item ID. To build an explainable recommendation solution, we combine the strengths of embedding-based and tree-based methods in a natural and effective manner, which to our knowledge has never been studied before.

2 PRELIMINARY

We first review the embedding-based model, discussing its difficulty in supporting explainable recommendation. We then introduce the tree-based model and emphasize its explanation mechanism.

2.1 Embedding-based Model

Embedding-based model is a typical example of representation learning [6], which aims to learn features from raw data for prediction. Matrix Factorization (MF) [26] is a simple yet effective embedding-based model for collaborative filtering, for which the predictive model can be formulated as:

$$\hat{y}_{MF}(u,i) = b_0 + b_u + b_i + \mathbf{p}_u^\top \mathbf{q}_i, \tag{1}$$

where b_0, b_u, b_i are bias terms, $\mathbf{p}_u \in \mathbb{R}^k$ and $\mathbf{q}_i \in \mathbb{R}^k$ are the embedding vector for user u and item i, respectively, and k denotes the embedding size.

In addition to IDs, users (items) are always associated with abundant side information, which may contain relevance signal of user preferences on items. Since most of these information are categorical variables, they are usually converted to real-valued feature vector via one-hot encoding [20, 32]. Let $\mathbf{x}_u$ and $\mathbf{x}_i$ denote the feature vector for user u and item i, respectively. To predict y_{ui}, a typical solution is to concatenate $\mathbf{x}_u$ and $\mathbf{x}_i$, *i.e.,* $\mathbf{x} = [\mathbf{x}_u, \mathbf{x}_i] \in \mathbb{R}^n$, which is then fed into a predictive model. FM [5, 32] is a representative of such predictive models, which is formulated as:

$$\hat{y}_{FM}(\mathbf{x}) = w_0 + \sum_{t=1}^{n} w_t x_t + \sum_{t=1}^{n} \sum_{j=t+1}^{n} \mathbf{v}_t^\top \mathbf{v}_j \cdot x_t x_j, \tag{2}$$

where w_0 and w_t are bias terms, $\mathbf{v}_t \in \mathbb{R}^k$ and $\mathbf{v}_j \in \mathbb{R}^k$ denote the embedding for feature t and j, respectively. We can see that FM associates each feature with an embedding, modeling the interaction of every two (nonzero) features via the inner product of their embeddings. If only user ID and item ID are used as the features of $\mathbf{x}$, FM can exactly recover the MF model; by feeding IDs and side features together into $\mathbf{x}$, FM models all pairwise (*i.e.,* second-order) interactions among IDs and side features.

With the recent advances of deep learning, neural network methods have also been employed to build embedding-based models [12, 20, 36]. Specially, Wide&Deep [12] and Deep Crossing [36] learn feature interactions by placing a multi-layer perceptron (MLP) above the concatenation of the embeddings of nonzero features; the MLP is claimed to be capable of learning any-order cross features. Neural FM (NFM) [20] first applies a bilinear interaction pooling on feature embeddings (*i.e.,* $\sum_{t=1}^{n} \sum_{j=t+1}^{n} x_t \mathbf{v}_t \odot x_j \mathbf{v}_j$) to learn second-order feature interactions, followed by a MLP to learn high-order features interactions.

Despite the strong representation ability of existing embedding-based methods in modeling side information, we argue that they are not suitable for providing explanations. FM models second-order feature interactions only and cannot capture high-order cross feature effects; moreover, it uniformly considers all second-order interactions and cannot distinguish which interactions are more important for a prediction [46]. While neural embedding models are able to capture high-order cross features, they are usually achieved by a nonlinear neural network above feature embeddings. The neural network stacks multiple nonlinear layers and is theoretically guaranteed to fit any continuous function [25], however, the fitting process is opaque and cannot be explained. To the best of our knowledge, there is no way to extract explicit cross features from the neural network and evaluate their contributions to a prediction.

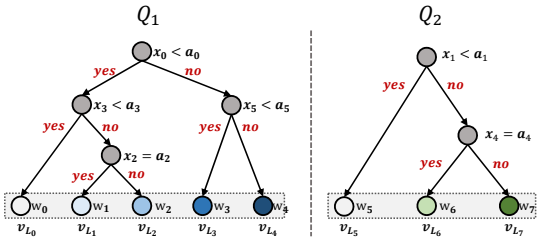


Figure 1: An example of a GBDT model with two subtrees.

2.2 Tree-based Model

In contrast to representation learning methods, tree-based models do not learn features for prediction. Instead, they perform prediction by learning decision rules from data. We represent the structure of a tree model as $Q = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V}$ and $\mathcal{E}$ denote the nodes and edges, respectively. The nodes in $\mathcal{V}$ have three types: the root node v_0, the internal (*aka.* decision) nodes $\mathcal{V}_T$, and the leaf nodes $\mathcal{V}_L$. Figure 1 illustrates an example of a decision tree model. Each decision node v_t splits a feature x_t with two decision edges: for numerical feature (*e.g.,* time), it chooses a threshold a_j and splits the feature into $[x_t < a_j]$ and $[x_t \geq a_j]$; for binary feature (*e.g.,* features after one-hot encoding on a categorical variable), it determines whether the feature equals to a value or not, *i.e.,* the decision edges are like $[x_t = a_j]$ and $[x_t \neq a_j]$.

A path from the root node to a leaf node forms a *decision rule*, which can also be seen as a *cross feature*, such as in Figure 1 the leaf node v_{L_2} represents $[x_0 < a_0]\&[x_3 \geq a_3]\&[x_2 \neq a_2]$. Each leaf node v_{L_i} has a value w_i, denoting the prediction value of the corresponding decision rule. Given a feature vector $\mathbf{x}$, the tree model first determines which leaf node $\mathbf{x}$ falls on, and then takes the value of the leaf node as the prediction: $\hat{y}_{DT}(\mathbf{x}) = w_{Q(\mathbf{x})}$, where Q maps the feature vector to the leaf node based on the tree structure. We can see that under such a prediction mechanism, the leaf node can be regarded as the most prominent cross feature for the prediction. As such, the tree-based model is self-interpretable by nature.

As one single tree may not be expressive enough to capture complex patterns in data, a more widely used solution is to build a forest, such as gradient boosting decision trees (GBDT) which boosts the prediction by leveraging multiple additive trees:

$$\hat{y}_{GBDT}(\mathbf{x}) = \sum_{s=1}^{S} \hat{y}_{DT_s}(\mathbf{x}), \tag{3}$$

where S denote the number of additive trees, and $\hat{y}_{DT_s}$ denotes the predictive model for the s-th tree. We can see that GBDT extracts S rules to predict the target value of a given feature vector, whereas a single tree model predicts based on one rule. As such, GBDT usually leads to better accuracy than a single tree model [7, 18].

While tree-based models are effective in generating interpretable predictions from rich side features, they suffer from generalizing to unseen feature interactions. As such, tree-based models cannot be used for collaborative filtering which needs to model the sparse ID features of users and items.

We can see that the pros and cons of embedding-based and tree-based models complement each other, in terms of generalization ability and interpretability. Hence, to build an effective and explainable recommender systems, a natural solution is to combine the two types of models.

3 TREE-ENHANCED EMBEDDING METHOD

We first present our tree-enhanced embedding method (TEM) that unifies the strengths of MF for sparse data modeling and GBDTs for cross feature learning. We then discuss the explainability and scrutability and analyze the time complexity of TEM.

3.1 Predictive Model

Given a user u, an item i, and their feature vectors $[\mathbf{x}_u, \mathbf{x}_i] = \mathbf{x} \in \mathbb{R}^n$ as the input, TEM predicts the user-item preference as,

$$\hat{y}_{TEM}(u, i, \mathbf{x}) = b_0 + \sum_{t=1}^{n} b_t x_t + f_\Theta(u, i, \mathbf{x}), \tag{4}$$

where the first two terms model the feature biases similar to that of FM, and $f_\Theta(u, i, \mathbf{x})$ is the core component of TEM with parameters Θ to model the cross feature effect, which is shown in Figure 2. In what follows, we elaborate the design of f_Θ step by step.

3.1.1 ***Constructing Cross Features.*** Instead of embedding-based methods that capture the cross feature effect opaquely during the learning process, our primary consideration is to make the cross features explicit and explainable. A widely used solution in industry is to manually craft cross features, and then feed them into an interpretable method that can learn the importance of each cross feature, such as logistic regression. For example, we can cross all values of feature variables *age* and *traveler style* to obtain the second-order cross features like *[age≥ 18] & [traveler style=friends]*. However, the difficulty of such method is that it is not scalable. For modeling higher-order feature interactions, one has to cross multiple feature variables together, resulting in exponential increase in complexity. With a large space of billions of features, even performing feature selection [43] is highly challenging, not to mention learning from them. Although through careful feature engineering such as crossing important variables or values only [12], one can control the complexity to a certain extent, it requires extensive domain knowledge to develop an effective solution and is not easily domain-adaptable.

To avoid such labor-intensive feature engineering, we leverage the GBDT (briefed in Section 2.2), to automatically identify useful cross features. While GBDT is not specially designed for extracting cross features, considering that a leaf node represents a cross feature and the trees are constructed by optimizing predictions on historical interactions, it is reasonable to think that the leaf nodes are useful cross features for prediction.

Formally, we denote a GBDT as a set of decision trees, $Q = \{Q_1, \cdots, Q_S\}$, where each tree maps a feature vector $\mathbf{x}$ to a leaf node (with a weight); we use L_s to denote the number of leaf nodes in the s-th tree. Distinct from the original GBDT that sums over the weights of activated leaf nodes as the prediction, we keep the activated leaf nodes as cross features, feeding them into a neural attention model for more effective learning. We represent the cross features as a multi-hot vector $\mathbf{q}$, which is a concatenation of multiple one-hot vectors (where a one-hot vector encodes the activated leaf node of a tree):

$$\mathbf{q} = GBDT(\mathbf{x}|Q) = [Q_1(\mathbf{x}), \cdots, Q_S(\mathbf{x})]. \tag{5}$$

Here $\mathbf{q}$ is a sparse vector, where an element of value 1 indicates an activated leaf node and the number of nonzero elements in $\mathbf{q}$ is S.

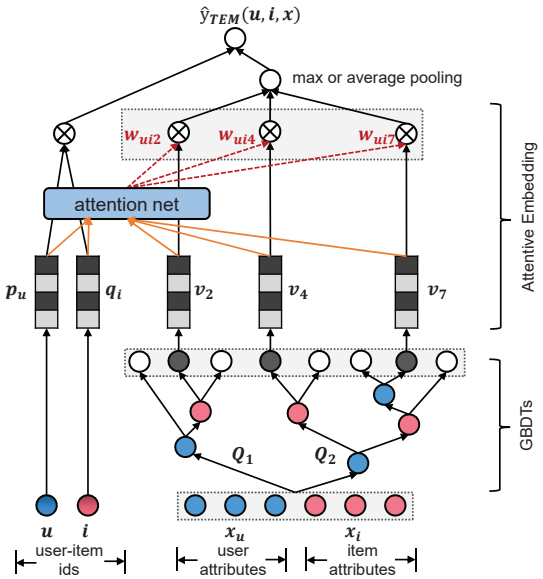


Figure 2: Illustrative architecture of our TEM framework.

Let the size of $\mathbf{q}$ be $L = \sum_s L_s$. For example, in Figure 1, there are two subtrees Q_1 and Q_2 with 5 and 3 leaf nodes, respectively. If $\mathbf{x}$ ends up with the second and third leaf node of Q_1 and Q_2, respectively, the resultant multi-hot vector $\mathbf{q}$ should be $[0, 1, 0, 0, 0, 0, 0, 1]$. Let the semantics of feature variables (x_0 to x_5) and values (a_0 to a_5) of Figure 1 be listed in Table 1, then $\mathbf{q}$ implies the two cross features extracted from $\mathbf{x}$:

(1) v_{L_1}: *[Age< 18] & [Country≠France] & [Restaurant Tag= French].*
(2) v_{L_7}: *[Expert Level≥ 4] & [Traveler Style≠Luxury Traveler].*

3.1.2 ***Prediction with Cross Features.*** With the explicit cross features, we can employ sparse linear methods to learn the importance of each cross feature, and select the top cross features as the explanation for a prediction. The prior work by Facebook [22] has demonstrated the effectiveness of such a solution, which feeds the leaf nodes of a GBDT into a logistic regression (LR) model. We term this solution as *GBDT+LR*. Although GBDT+LR is capable of learning the importance of cross features, it assigns a cross feature the same weight for predictions of all user-item pairs, which limits the modeling fidelity. In real applications, it is common that users with similar demographics may choose similar items, but they are driven by different intents or reasons.

As an example, let $(u, i, \mathbf{x})$ and $(u', i', \mathbf{x}')$ be two positive instances. Assuming $\mathbf{x}$ equals to $\mathbf{x}'$, then the two instances will have the same cross features from GBDT. Since each cross feature has a global weight independent of the training instance in LR, the predictions of (u, i) and (u', i') will be interpreted as the same top cross features, regardless of the possibility that the actual reasons behind u chose i and u' chose i' are different. To ensure the expressiveness, we believe it is important to score the cross features differently for different user-item pairs, *i.e.,* personalizing the weights on cross features rather than using a global weighting mechanism.

Recent advances on neural recommender models such as Wide&Deep [12] and NFM [20] can allow personalized importance on cross features. This is achieved by embedding user ID, item ID, and cross features together into a shared embedding space, and

Table 1: The semantics of feature variables and values of the GBDT model in Figure 1.

$x_0 \leftarrow$ *Age*	$x_1 \leftarrow$ *Expert Level*	$x_2 \leftarrow$ *Restaurant Tag*
$a_0 \leftarrow 18$	$a_1 \leftarrow 4$	$a_2 \leftarrow$ *French*
$x_3 \leftarrow$ *Country*	$x_4 \leftarrow$ *Traveler Style*	$x_5 \leftarrow$ *Price*
$a_3 \leftarrow$ *France*	$a_4 \leftarrow$ *Luxury Traveler*	$a_5 \leftarrow$ *$$$$*

then performing nonlinear transformations (*e.g.,* by fully connected layers) on the embedding vectors. The strong representation power of nonlinear hidden layers enables complicated interactions among user ID, item ID, and cross features to be captured. As such, a cross feature can impact differently when predicting with different user-item pairs. However, such methods cannot interpret the personalized weights of cross features, due to the hardly explainable nonlinear hidden layers. As such, for explainability purpose we have to discard the use of fully connected hidden layers, although they are helpful to a model's performance in existing methods.

To develop a method that is both effective and explainable, we introduce two essential ingredients of our TEM — embedding and attention. Specifically, we first associate each cross feature with an embedding vector, allowing the correlations among cross features to be captured. We then devise an attention mechanism to explicitly model the personalized weights on cross features. Lastly, the embeddings of user ID, item ID, and cross features are integrated together for the final prediction. The use of embedding and attention endows TEM strong representation ability and guarantees the effectiveness, even though it is a shallow model without any fully connected hidden layer. In what follows, we elaborate the two key ingredients of TEM.

Embedding. Given the cross feature vector $\mathbf{q}$ generated by GBDT, we project each cross feature j into an embedding vector $\mathbf{v}_j \in \mathbb{R}^k$, where k is the embedding size. After the operation, we obtain a set of embedding vectors $\mathcal{V} = \{q_1\mathbf{v}_1, \cdots, q_L\mathbf{v}_L\}$. Since $\mathbf{q}$ is a sparse vector with only a few nonzero elements, we only need to include the embeddings of nonzero features for a prediction, *i.e.,* $\mathcal{V} = \{\mathbf{v}_l\}$ where $q_l \neq 0$. We use $\mathbf{p}_u$ and $\mathbf{q}_i$ to denote the user embedding and item embedding, respectively.

There are two advantages of embedding the cross features into a vector space, compared to LR that uses a scalar to weight a feature. First, learning with embeddings can capture the correlations among features, *e.g.,* frequently co-occurred features may yield similar embeddings, which can alleviate the data sparsity issue. Second, it provides a means to seamlessly integrate the output of GBDT with the embedding-based collaborative filtering, being more flexible than a late fusion on the model predictions (*e.g.,* boosting GBDT with FM as used in [49]).

Attention. Inspired by the previous work [9, 46], we explicitly capture the varying importance of cross features on prediction by assigning an attentive weight for the embedding of each cross feature. Here we consider two ways to aggregate the embeddings of cross features, average pooling and max pooling, to obtain a unified representation $\mathbf{e}(u, i, \mathcal{V})$ for cross features:

$$\begin{cases} \mathbf{e}_{avg}(u,i,\mathcal{V}) = \frac{1}{|\mathcal{V}|}\sum_{\mathbf{v}_l \in \mathcal{V}} w_{uil}\mathbf{v}_l, \\ \mathbf{e}_{max}(u,i,\mathcal{V}) = max\_pool_{\mathbf{v}_l \in \mathcal{V}}(w_{uil}\mathbf{v}_l), \end{cases} \tag{6}$$

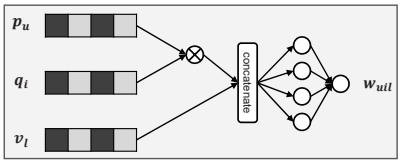


Figure 3: Illustration of the attention network in TEM.

where w_{uil} is a trainable parameter denoting the attentive weight of the l-th cross feature in constituting the unified representation, and importantly, it is personalized to be dependent with (u, i).

While the above solution seems to be sound and explainable, the problem is that for (u, i) pairs that have never co-occurred before, the attentive weight w_{uil} cannot be estimated. In addition, the parameter space of w is too large — there are UIL weights in total (where U, I, and L denote the number of users, items, and the size of $\mathbf{q}$, respectively), which is impractical to materialize for real-world applications. To address the generalization and scalability issues, we consider modeling w_{uil} as a function dependent on the embeddings of u, i, and l, rather than learning w_{uil} freely from data. Inspired by the recent success [4, 9, 46] that uses multi-layer perceptrons (MLPs) to learn the attentive weights, we similarly use a MLP to parameterize w_{uil}. We call the MLP as the *attention network*, which is defined as:

$$\begin{cases} w'_{uil} & = \mathbf{h}^\top ReLU(\mathbf{W}\,([\mathbf{p}_u \odot \mathbf{q}_i, \mathbf{v}_l]) + \mathbf{b}) \\ w_{uil} & = \frac{exp(w'_{uil})}{\sum_{(u,i,\mathbf{x})\in O} exp(w'_{uil})} \end{cases}, \tag{7}$$

where $\mathbf{W} \in \mathbb{R}^{a\times 2k}$ and $\mathbf{b} \in \mathbb{R}^a$ denote the weight matrix and bias vector of the hidden layer, respectively, and a controls the size of the hidden layer. The vector $\mathbf{h} \in \mathbb{R}^a$ projects the hidden layer into the attentive weight for output. We used the rectifier as the activation function and normalized the attentive weights using softmax. Figure 3 illustrates the architecture of our attention network, and we term a as the *attention size*.

Final Prediction. Having established the attentive embeddings, we obtain a unified embedding vector $\mathbf{e}(u, i, \mathcal{V})$ for cross features. To incorporate the CF modeling, we concatenate $\mathbf{e}(u, i, \mathcal{V})$ with $\mathbf{p}_u \odot \mathbf{q}_i$, which reassembles MF to model the interaction between user ID and item ID. We then apply a linear regression to project the concatenated vector to the final prediction. This leads to the predictive model of our TEM as:

$$\hat{y}_{TEM}(u, i, \mathbf{x}) = b_0 + \sum_{t=1}^{m} b_t x_t + \mathbf{r}_1^\top(\mathbf{p}_u \odot \mathbf{q}_i) + \mathbf{r}_2^\top \mathbf{e}(u, i, \mathcal{V}), \tag{8}$$

where $\mathbf{r}_1 \in \mathbb{R}^k$ and $\mathbf{r}_2 \in \mathbb{R}^k$ are the weights of the final linear regression layer. As can be seen, our TEM is a shallow and additive model. To interpret a prediction, we can easily evaluate the contribution of each component. We use **TEM-avg** and **TEM-max** to denote the TEM that uses $\mathbf{e}_{avg}(\cdot)$ and $\mathbf{e}_{max}(\cdot)$, respectively, and discuss their explanation schemes in Section 3.3.1.

3.2 Learning

Similar to the recent work on neural collaborative filtering [21], we solve the item recommendation task as a binary classification problem. Specifically, an observed user-item interaction is assigned to a target value 1, otherwise 0. We optimize the pointwise log loss, which forces the prediction score $\hat{y}_{ui}$ to be close to the target y_{ui}:

$$\mathcal{L} = \sum_{(u,i,\mathbf{x})\in O} -y_{ui} \log \sigma(\hat{y}_{ui}) - (1 - y_{ui}) \log (1 - \sigma(\hat{y}_{ui})), \tag{9}$$

where σ is the activation function to restrict the prediction to be in $(0, 1)$, set as sigmoid $\sigma(x) = 1/(1 + e^{-x})$ in this work. The regularization terms are omitted here for clarity (we tuned the L_2 regularization in experiments when overfitting was observed). Note that optimizing other objective functions are also technically viable, such as the pointwise regression loss [20, 41, 42] and ranking loss [9, 33, 44]. In this work, we use the log loss as a demonstration of our TEM.

Since TEM consists of two cascaded models, both them are trained to optimize the same log loss. We first train the GBDT, which greedily fits additive trees on the whole training data [10]. After obtaining the cross features from GBDT, we optimize the embedding-based prediction model using the mini-batch Adagrad [16]. Each mini-batch contains stochastic positive instances and randomly paired negative instances. Same as the optimal setting of [21], we pair one positive instance with four negative instances, which empirically shows good performance.

3.3 Discussion

3.3.1 ***Explainability & Scrutability.*** The two pooling methods as defined in Equation (6) aggregate the embeddings of cross features differently, resulting in different explanation mechanisms for TEM-avg and TEM-max. Specifically, the average pooling linearly combines all embeddings, with each embedding a weight to denote its importance. As such, the w_{uil} of $\mathbf{e}_{avg}(u, i, \mathcal{V})$ can be directly used to select top cross features (*i.e.,* decision rules) as the explanation of a prediction [4, 46]. In contrast, the max pooling is a nonlinear operator, where the d-th dimension of $\mathbf{e}_{max}(u, i, \mathcal{V})$ is set to be that of the l-th cross feature embedding with the maximum $w_{uil}v_{ld}$. As such, at most k cross feature embeddings will contribute to the unified representation[1], and we can treat the max pooling as performing feature selection on cross features in the embedding space. To select top cross features for explanation, we need to track the embeddings of which cross features contribute most during the max pooling, rather than simply relying on w_{uil}. We conduct a case study on explainability of TEM in Section 4.4.1.

Empowered by the transparency in generating a recommendation, TEM allows the recommender to be scrutable [39]. If a user is unsatisfied with a recommendation due to improper reasons, TEM allows a user to correct the reasoning process to obtain refreshed recommendations. As Equation (8) shows, we can easily obtain the contribution of each cross feature on the final prediction, *e.g.,* $y_{uil} = w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$ for TEM-avg. When getting feedback from a user (*i.e.,* the signals indicating what she likes or not), we can localize the cross features that contain the signals, and then modify the corresponding attentive weights. As such, we can refresh the predictions and re-rank the recommendation list without re-training the whole model. We use a case study to demonstrate the scrutability of TEM in Section 4.4.2.

[1]Typically, the embedding size k is smaller than the number of trees S in GBDT.

3.3.2 ***Time Complexity Analysis.*** As we separate the learning procedure into two phases, we can calculate the computational costs step by step. Generally, the time complexity of building a GBDT model is $O(SD\,\|\mathbf{x}\|_0 \log n)$, where S is the number of trees, D is the maximum depth of trees, n is the number of training instances, and $\|\mathbf{x}\|_0$ denotes the average number of non-zero entries in the training instances. Moreover, we can speed up the greedy algorithm in GBDT by using the block structure like XGBoost [10].

For the embedding component, calculating the attention score for each (u, i, l) costs time $O(2ak)$, where a and k are the attention and embedding size, respectively. Accordingly, adopting the pooling operation for each (u, i) costs $O(2akS)$. As such, to train the embedding model of TEM over n training instances, the complexity is $O(2akSn)$. Therefore, the overall time complexity for training TEM from scratch is $O(SD\,\|\mathbf{x}\|_0 \log n + 2akSn)$.

4 EXPERIMENTS

As the key contribution of the work is to generate accurate and explainable recommendations, we conduct experiments to answer the following questions:

(1) **RQ1**: Compared with the state-of-the-art recommendation methods, can our TEM achieve comparable accuracy?
(2) **RQ2**: Can TEM make the recommendation results easy-to-interpret by using cross features and the attention network?
(3) **RQ3**: How do different hyper-parameter settings (*e.g.,* the number of trees and embedding size) affect TEM?

4.1 Data Description

We collect data from two populous cities in TripAdvisor[2], London (LON) and New York City (NYC), and separately perform experiments of tourist attraction and restaurant recommendation. We term the two datasets as LON-A and NYC-R respectively. In particular, we crawl 1,001 tourist attractions (*e.g., British Museum*) from LON with the corresponding ratings written by 17,238 users from August 2014 to August 2017; similarly, 8,791 restaurants (*e.g., The River Cafe*) and 16,015 users are obtained from NYC. The ratings are transformed into binary implicit feedback as ground truth, indicating whether the user has interacted with the specific item. To ensure the quality of the data, we retain users/items with at least five ratings only. The statistics of two datasets are summarized in Table 2. Moreover, we have collected the natural or system-generated labels that are affiliated with users and items as their side information (*aka.* profile). Particularly, the profile of each user includes gender (*e.g., Female*), age (*e.g.,* 25-34), and traveler styles (*e.g., Foodie* and *Beach Goer*); meanwhile, the side information of an item consists of attributes (*e.g., Art Museum* and *French*), tags (*e.g., Rosetta Stone* and *Madelenies*), and price (*e.g., $$$*). We have summarized all types of user/item side information in Table 3.

For each dataset, we holdout the latest 20% interaction history of each user to construct the test set, and randomly split the remaining data into training (70%) and validation (10%) sets. The validation set is used to tune hyper-parameters and the final performance comparison is conducted on the test set.

[2]https://www.tripadvisor.com.

Table 2: Statistics of the datasets.

Dataset	User#	User Feature#	Item#	Item Feature#	Interaction#
LON-A	16,315	3,230	953	4,731	136,978
NYC-R	15,232	3,230	6,258	10,411	129,964

Table 3: Statistics of the side information, where the dimension of each feature is shown in parentheses.

Side Information	Features (Category#)
LON-A/NYC-R User Feature	Age (6), Gender (2), Expert Level (6), Traveler Styles (18), Country (126), City (3,072)
LON-A Attraction Feature	Attributes (89), Tags (4,635), Rating (7)
NYC-R Restaurant Feature	Attributes (100), Tags (10,301), Price (3), Rating (7)

4.2 Experimental Settings

4.2.1 ***Evaluation Protocols.*** Given one positive user-item interaction in the testing set, we pair it with 50 negative instances that the user did not consume before. Then each method outputs prediction scores for these 51 instances. To evaluate the prediction scores, we adopt two metrics: the error-based log loss and the ranking-aware ndcg@K.

- **logloss**: logarithmic loss [36] measures the probability that one predicted user-item interaction diverges from the ground truth. A lower logloss indicates a better performance.
- **ndcg@K**: ndcg@K [17, 19, 21, 29, 30] assigns the higher importance to the items within the top K positions of the ranking list. A higher ndcg@K reflects a better accuracy of getting top ranks correct.

We report the average scores for all testing instances, where logloss indicates the generalization ability of each model, and ndcg reflects the performance for top-K recommendation. The same settings apply for the hyper-parameter tuning on the validation set.

4.2.2 ***Baselines.*** We compare our TEM with the following methods to justify the rationality of our proposal:

- **XGBoost** [10]: This is the state-of-the-art tree-based method that captures complex feature dependencies (*aka.* cross features).
- **GBDT+LR** [22]: This method feeds the cross features extracted from GBDT into the logistic regression, aiming to refine the weights for each cross feature.
- **GB-CENT** [49]: Such state-of-the-art boosting method combines the prediction results from MF and GBDT. To adjust GB-CENT to perform our tasks, we input the ID features and side information to MF and GBDT, respectively.
- **FM** [32]: This is a generic embedding-based model that encodes side information and IDs with embedding vectors. It implicitly models all the second-order cross features via the inner product of any two feature embeddings.
- **NFM** [20]: Neural FM is the state-of-the-art factorization model under the neural network framework. It stacks multiple fully connected layers above the inner products of feature embeddings to capture higher-order and nonlinear cross features. Specially, we employed one hidden layers for NFM as suggested in [20].

4.2.3 ***Parameter Settings.*** For a fair comparison, we optimize all the methods with the same objective function of Equation (9). We implement our proposed TEM[3] using Tensorflow[4]. We use

[3]https://github.com/xiangwang1223/TEM.
[4]https://www.tensorflow.org.

Table 4: Performance comparison between all the methods, where the significance test is based on logloss of TEM-max.

Dataset	LON-A			NYC-R		
Method	logloss	ndcg@5	p-value	logloss	ndcg@5	p-value
XGBoost	0.1251	0.6785	8e−5	0.1916	0.3943	4e−5
GBDT+LR	0.1139	0.6790	2e−4	0.1914	0.3997	4e−4
GB-CENT	0.1246	0.6784	6e−5	0.1918	0.3995	4e−5
FM	0.0939	0.6809	1e−2	0.1517	0.4018	5e−5
NFM	0.0892	0.6812	2e−2	0.1471	0.4020	8e−4
TEM-avg	0.0818	0.6821	–	0.1235	0.4019	–
TEM-max	**0.0791**	**0.6828**	–	**0.1192**	**0.4038**	–

XGBoost[5] to implement the tree-based components of all methods, where the number of trees and the maximum depth of trees is searched in $\{100, 200, 300, 400, 500\}$ and $\{3, 4, 5, 6\}$, respectively. For all embedding-based components, we test the embedding size of $\{5, 10, 20, 40\}$, and empirically set the attention size same as the embedding size. All embedding-based methods are optimized using the mini-batch Adagrad for a fair comparison, where the learning rate is searched in $\{0.005, 0.01, 0.05, 0.1, 0.5\}$. Moreover, the early stopping strategy is performed, where we stopped training if the logloss on the validation set increased for four successive epoches. Without special mention, we show the results of tree number 500, maximum depth 6, and embedding size 20, and more results of the key parameters are shown in Section 4.5.

4.3 Performance Comparison (RQ1)

We start by comparing the performance of all the methods. We then explore how the cross features affect the recommendation results.

4.3.1 ***Overall Comparison.*** Table 4 displays the performance comparison *w.r.t.* logloss and ndcg@5 on LON-A and NYC-R datasets. We have the following observations:

- XGBoost achieves poor performance since it treats sparse IDs as ordinary features and hardly derives useful cross features based on the sparse data. It hence fails to capture the collaborative filtering effect. Moreover, it cannot generalize to unseen feature dependencies. GBDT+LR slightly outperforms XGBoost, verifying the feasibility of treating cross features as the input of one classifier and revising the weight of each cross feature.
- The performance of GB-CENT indicates that such boosting may be insufficient to fully facilitate information propagation between two models. Note that to reduce the computational complexity, the modified GB-CENT only conducts GBDT over all the instances, rather than performing GBDT over the supporting instances of each categorical feature as suggested in [49]. Such modification may contribute to the unsatisfactory performance.
- When performing our recommendation tasks, FM and NFM, outperform XGBoost, GBDT+LR, and GB-CENT. It is reasonable since they are good at modeling the sparse interactions and the underlying second-order cross features. NFM benefits from the higher-order and nonlinear feature correlations by leveraging neural networks, thus leads to better performance than FM.
- TEM achieves the best performance, substantially outperforming NFM *w.r.t.* logloss and obtaining a comparable ndcg@5. By integrating the embeddings of cross features, TEM can achieve the comparable expressiveness to NFM. While NFM treats all feature interactions equally, TEM can employ the attention networks on identifying the personalized attention of each cross feature. We further conduct one-sample t-tests to verify that all improvements are statistically significant with p-value < 0.05.

4.3.2 ***Effect of Cross Features.*** To analyze the effect of cross features, we consider the variants that remove cross feature modeling, termed as FM-c, NFM-c, TEM-avg-c, and TEM-max-c. For FM and NFM, one user-item interaction is represented only by the sum of the user and item ID embeddings and their attribute embeddings, without any interactions among features. For TEM, we skip the cross feature extraction and direct feed into the raw features. As shown in Figure 4, we have the following findings:

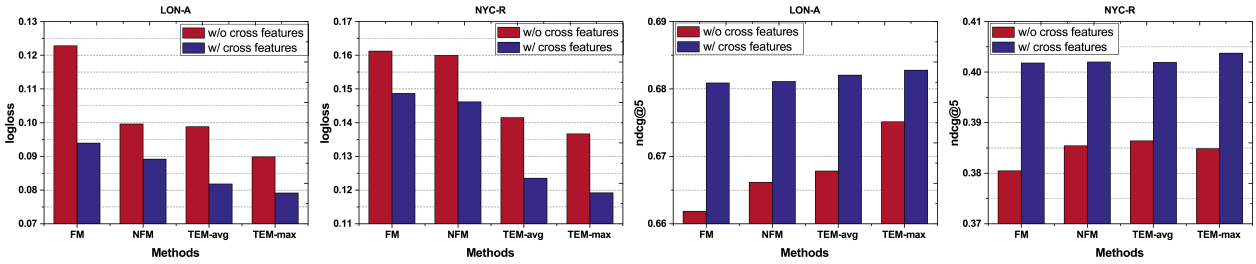
- For all methods, removing cross feature modeling hurts the expressiveness adversely and degrades the recommendation performance. FM-c and NFM-c assume one user/item and her/its attributes are linearly independent, which fail to encode any interactions between them in the embedding space. Taking advantages of the attention network, TEM-avg-c and TEM-max-c still model the interactions between IDs and attributes, and achieve better representation ability than FM-c and NFM-c.
- As Figures 4(a) and 4(b) demonstrate, TEM significantly outperforms FM and NFM by a large margin *w.r.t.* logloss, verifying the substantial influence of explicit cross feature modeling. While FM and NFM consider all the underlying feature correlations, neither of them explicitly presents the cross features or identifies the importance of each cross feature. This makes them work as a black-box and hurts their explainability. Therefore, the improvement achieved by TEM again verifies the effectiveness of the explicit cross features refined from the tree-based component.
- Lastly, while exhibiting the lowest logloss, TEM achieves only comparable performance *w.r.t.* ndcg@5 to that of NFM, as shown in Figures 4(c) and 4(d). It indicates the unsatisfied generalization ability of TEM, since the cross features extracted from GBDT only reflect the feature dependencies observed in the dataset and consequently TEM cannot generalize to the unseen rules. We leave the further exploration of the generalization ability of our TEM as the future work.

4.4 Case Studies (RQ2)

Apart from being comparable at predictive accuracy, the key advantage of TEM over other methods is that its learning process is transparent and easily explainable. Towards this end, we show examples drawn from TEM-avg on LON-A to demonstrate its explainability and scrutability.

4.4.1 ***Explainability.*** To demonstrate the explainability of TEM, we focus on a sampled user, whose profile is {age: 35-49, gender: *female*, country: *the United Kingdom*, city: *London*, expert level: 4, traveler styles: *Art and Architecture Lover, Peace and Quite Seeker, Family Vacationer, Urban Explorer*}; meanwhile, we randomly select five attractions, {i_{31}: *National Theatre*, i_{45}: *The View form the Shard*, i_{49}: *The London Eye*, i_{93}: *Camden Street Art Tours*, i_{100}: *Royal opera House*}, from the user's holdout testing set. Figure 5 visualizes the learning results, where a row represents an attraction, and a

[5]https://xgboost.readthedocs.io.



(a) logloss on LON-A (b) logloss on NYC-R (c) ndcg@5 on LON-A (d) ndcg@5 on NYC-R

Figure 4: Performance comparison of logloss *w.r.t.* the cross features on LON-A and NYC-R datasets.

column represents a cross feature (we sample five cross features which are listed in Table 5). The left heat map presents her attention scores over the five sampled cross features and the right displays the contributions of these cross features for the final prediction.

We first focus on the left heat map of attention scores. Examining the attention scores of a row, we can explain the recommendation for the corresponding attraction using the top cross features. For example, we recommend *The View from the Shard* (*i.e.,* the second row i_{45}) for the user mainly because of the dominant cross feature v_{130}, evidenced by the highest attention score of 1 (*cf.* the entry at the second row and the third column). Based on the attention scores, we can attribute her preferences on *The View from the Shard* to her special interests in the item aspects of *Walk Around* (from v_{130}), *Top Deck & Canary Wharf* (from v_{22}), and *Camden Town* (from v_{148}). To justify the rationality of the reasoning, we further check the user's visiting history, finding that the three item aspects have frequently occurred in her historical items.

In right heat map of Figure 5, an entry denotes the contribution of the corresponding cross feature (*i.e.,* $y'_{uil} = w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$) to the final prediction Jointly analyzing the left and right heat maps, we find that the attention score w_{uil} is generally consistent with y_{uil}, which contains useful cues about the user's preference. Based on

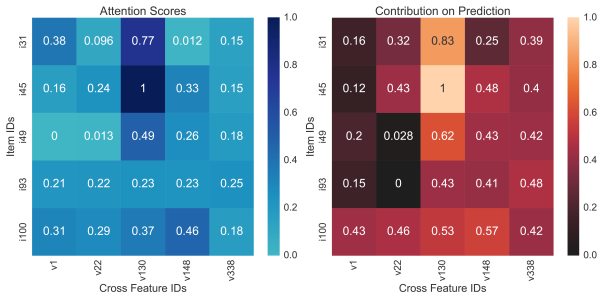


Figure 5: Visualization of cross feature attentions produced by TEM-avg on LON-A. An entry of the left and right heat map visualizes the attention value w_{uil} and its contribution to the final prediction, *i.e.,* $w_{uil}\mathbf{r}_2^\top \mathbf{v}_l$, respectively.

Table 5: Descriptions of the cross features in Figure 5.

ID	Description of Cross Features shown in Figure 5
v_1	[User Country=UK] & [User Style=Art and Architecture Lover] ⇒ [Item Attribute=Concerts and Shows] & [Item Tag=Imelda Staunton]
$\mathbf{v_{22}}$	[User Age=35-49] & [User Country=UK] ⇒ [Item Tag=**Camden Town**] & [Item Rating=4.0]
$\mathbf{v_{130}}$	[User Age≠ 25-34] & [User Gender=Female] & [User Style=Peace and Quiet Seeker] ⇒ [Item Attribute=Sights & Landmarks] & [Item Tag=**Walk Around**]
$\mathbf{v_{148}}$	[User Age≠ 50-64] & [User Country≠USA] ⇒ [Item Tag=**Top Deck & Canary Wharf**]
v_{336}	[User Age=35-49] & [User Country=UK] & [User Style=Art and Architecture Lover] ⇒ [Item Tag=Royal Opera House] & [Item Tag=Interval Drinks]

such outcome, we can utilize the attention scores of cross features to explain a recommendation (*e.g.,* the user prefers i_{45} owing to the top rules of v_{130} and v_{148} weighted with personalized attention scores of 1 and 0.33). This case demonstrates TEM's capability of providing more informative explanations according to a user's preferred cross features, which we believe are better than mere labels or similar user/item list.

4.4.2 ***Scrutability.*** Apart from making the recommendation process transparent, our TEM can further allow a user to correct the process, so as to refresh the recommendation as she desires. This property of adjusting recommendation is known as the scrutability [19, 39]. As for TEM, the attention scores of cross features serve as a gateway to exert control on the recommendation process. We illustrate it using another sampled user in Table 6.

The profile of this user indicates that she enjoys the traveler style of *Urban Explorer* most; moreover, most attractions in the historical interactions of her are tagged with *Sights & Landmarks*, *Points of Interest* and *Neighborhoods*. Hence, TEM detects such frequent co-occurred cross features and accordingly recommends some attractions like *Old Compton Street* and *The Mall* to her. Assuming that the user attempts to scrutinize TEM and would like to visit some attractions tagged with *Garden* that are suitable for the *Nature Lover*. Towards this end, we assign the cross features containing *[User Style=Nature Lover] & [Item Attribute=Garden]* with a higher attentive weight, and then get the predictions of TEM to refresh the recommendations. In the adjusted recommendation list, the *Greenwich Foot Tunnel*, *Covent Garden*, and *Kensington Gardens* are ranked at the top positions. Therefore, based on the transparency and simulated scrutability, we believe that our TEM is easy-to-interpret, explainable and scrutable.

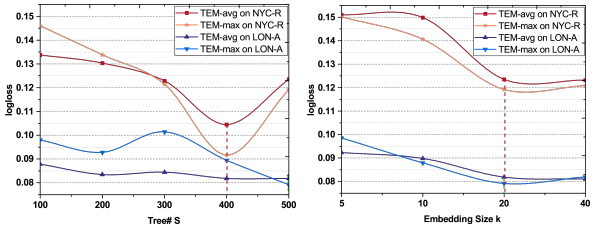
4.5 Hyper-parameter Studies (RQ3)

We empirically study the influences of several factors, such as the number of trees and the embedding size, on our TEM method.

4.5.1 ***Impact of Tree Number.*** The number of trees in TEM indicates the coverage of cross features, reflecting how much useful

Table 6: Scrutable recommendation for a sampled user on LON-A, where the first row and second row list the original and adjusted recommended attractions, respectively.

	Top Ranked Recommendation List on LON-A
1. Original	1. London Fields Park, 2. Old Compton Street, 3. The Mall, 4. West End, 5. Millennium Bridge
2. Adjusted	1. London Fields Park, 2. Greenwich Foot Tunnel, 3. Covent Garden, 4. Kensington Gardens, 5. West End



(a) logloss vs. tree number S (b) logloss vs. embedding size k

Figure 6: Performance comparison of logloss *w.r.t.* the tree number S and the embedding size k.

information is derived from the datasets. Figure 6(a) presents the performance *w.r.t.* logloss by varying the tree number S. We can see the logloss of TEM gradually decreases with more trees, whereas the performance is generally improved. Using a tree number of 400 and 500 leads to the best performance on NYC-R and LON-A, respectively. When the tree number exceeds the optimal settings (*e.g.,* S equals to 500 on NYC-R), the logloss increases, which may suffer from overfitting. This emphasizes the significance of the tree settings, which is consistent with [22, 49]

4.5.2 ***Impact of Embedding Size.*** The empirical results displayed in Figure 6(b) indicates the substantial influence of embedding size upon TEM. Enlarging the embedding size, TEM benefits from more powerful representations of the user-item pairs. Moreover, TEM-max shows consistent improvement over TEM-avg in most cases. We attributed such improvement to the nonlinearity achieved by the max pooling operation, which can select most informative cross features out, as discussed in Section 3.1.2. However, the oversized embedding may cause overfitting and degrade the performance, which is consistent with [20, 44]

5 RELATED WORK

We can roughly divide explanation styles into similarity-based and content-based categories. The similarity-based methods [1, 2] present explanations as a list of most similar users or items. For example, Behnoush *et al.* [1] used Restricted Boltzmann Machines to compute the explainability scores of the items in the top-K recommendation list. While the similarity-based explanation can serve as a generic solution for explaining a CF recommender, the drawback is that it lacks concrete reasoning.

Content-based works have considered various side information, ranging from item tags [38, 40], social relationships [31, 37], contextual reviews written by users [13, 15, 28, 31, 48] to knowledge graphs [3, 8, 47].

Item Tags. To explain a recommendation, the work [40] considered the matching between the relevant tags of an item and the preferred tags of the user.

Social Relations. Considering the user friendships in social networks, [37] proposed a generative model to investigate the effects of social explanations on user preferences.

Contextual Reviews. Zhang *et al.* [48] developed an explicit factor model, which incorporated user sentiments *w.r.t.* item aspects as well as the user-item ratings, to facilitate generating aspect-based explanations. Similarly, He *et al.* [19] extracted item aspects from user reviews and modeled the user-item-aspect relations in a hybrid collaborative filtering model. More recently, Ren *et al.* [31] involved the viewpoints, a tuple of user sentiment and item aspect, and trusted social relations in a latent factor model to boost recommendation performance and present personalized viewpoints as explanations.

Knowledge Graphs. Knowledge graphs show great potential on explainable recommendation. Yu *et al.* [47] introduced a meta-path-based factor model that paths learned from an information graph can enhance the user-item relations and further provide explainable reasoning. Recently, Alashkar *et al.* [3] integrated domain knowledge represented as logic-rules with the neural recommendation method.

Despite the promising attempts achieved, most previous works treat the extracted feature (*e.g.,* item aspect, user sentiment, or relationship) as an individual factor in factor models, same as the IDs. As such, little attention has been paid to discover the effects of cross features (or feature combinations) explicitly.

In terms of techniques, existing works have also considered combining tree-based and embedding-based models, among which the most popular method is boosting [11, 27, 49]. These solutions typically perform a late fusion on the prediction of two kinds of models. GB-CENT proposed in [49] composes of embedding and tree components to achieve the merits of both models. Particularly, GB-CENT achieves CF effect by conducting MF over categorical features; meanwhile, it employs GBDT on the supporting instances of numerical features to capture the nonlinear feature interactions. Ling *et al.* [27] shows that boosting neural networks with GBDT achieves the best performance in the CTR prediction. However, these boosting methods only fuse the outputs of different models and may be insufficient to fully propagate information between tree-based and embedding-based models. Distinct from the previous works, our TEM treats the cross features extracted from GBDT as the input of embedding-based model, facilitating the information propagation between two models. More importantly, the main focus of TEM is to provide explanations for a recommendation, rather than only for improving the performance.

6 CONCLUSION

In this work, we proposed a tree-enhanced embedding method (TEM), which seamlessly combines the generalization ability of embedding-based models with the explainability of tree-based models. Owing to the explicit cross features extracted from tree-based part and the easy-to-interpret attention network, the whole prediction process of our solution is fully transparent and self-explainable. Meanwhile, TEM can achieve comparable performance as the state-of-the-art recommendation methods.

In future, we will extend our TEM in three directions. First, we attempt to jointly learn the tree-based and embedding-based models, rather than separately modelling two components. This can facilitate the information propagation between two components. Second, we consider other context information, such as time, location, and user sentiments, to further enrich our explainability. Third, we will explore the effectiveness of involving knowledge graphs and logic rules into our TEM.

Acknowledgement This research is part of NExT++ project, supported by the National Research Foundation, Prime Minister's Office, Singapore under its IRC@Singapore Funding Initiative.

REFERENCES

[1] Behnoush Abdollahi and Olfa Nasraoui. 2016. Explainable Restricted Boltzmann Machines for Collaborative Filtering. (2016).

[2] Behnoush Abdollahi and Olfa Nasraoui. 2017. Using Explainability for Constrained Matrix Factorization. In *RecSys*. 79–83.

[3] Taleb Alashkar, Songyao Jiang, Shuyang Wang, and Yun Fu. 2017. Examples-Rules Guided Deep Neural Network for Makeup Recommendation. In *AAAI*. 941–947.

[4] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *ICLR*.

[5] Immanuel Bayer, Xiangnan He, Bhargav Kanagal, and Steffen Rendle. 2017. A Generic Coordinate Descent Framework for Learning from Implicit Feedback. In *WWW*. 1341–1350.

[6] Y. Bengio, A. Courville, and P. Vincent. 2013. Representation Learning: A Review and New Perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 8 (2013), 1798–1828.

[7] Leo Breiman. 2001. Random Forests. *Machine Learning* 45, 1 (2001), 5–32.

[8] Rose Catherine and William W. Cohen. 2016. Personalized Recommendations using Knowledge Graphs: A Probabilistic Logic Programming Approach. In *RecSys*. 325–332.

[9] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive Collaborative Filtering: Multimedia Recommendation with Item- and Component-Level Attention. In *SIGIR*. 335–344.

[10] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. In *SIGKDD*. 785–794.

[11] Tianqi Chen, Linpeng Tang, Qin Liu, Diyi Yang, Saining Xie, Xuezhi Cao, Chunyang Wu, Enpeng Yao, Zhengyang Liu, Zhansheng Jiang, et al. 2012. Combining factorization model and additive forest for collaborative followee recommendation. *KDD CUP* (2012).

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *DLRS*. 7–10.

[13] Zhiyong Cheng, Ying Ding, Lei Zhu, and Mohan Kankanhalli. 2018. Aspect-Aware Latent Factor Model: Rating Prediction with Ratings and Reviews. In *WWW*.

[14] Zhiyong Cheng and Jialie Shen. 2016. On Effective Location-Aware Music Recommendation. *TOIS* 34, 2 (2016), 13:1–13:32.

[15] Qiming Diao, Minghui Qiu, Chao-Yuan Wu, Alexander J. Smola, Jing Jiang, and Chong Wang. 2014. Jointly modeling aspects, ratings and sentiments for movie recommendation (JMARS). In *KDD*. 193–202.

[16] John C. Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *JMLR* 12 (2011), 2121–2159.

[17] Fuli Feng, Xiangnan He, Yiqun Liu, Liqiang Nie, and Tat-Seng Chua. 2018. Learning on Partial-Order Hypergraphs. In *WWW*.

[18] Jerome H Friedman. 2001. Greedy function approximation: a gradient boosting machine. *Annals of statistics* (2001), 1189–1232.

[19] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. TriRank: Review-aware Explainable Recommendation by Modeling Aspects. In *CIKM*. 1661–1670.

[20] Xiangnan He and Tat-Seng Chua. 2017. Neural Factorization Machines for Sparse Predictive Analytics. In *SIGIR*. 355–364.

[21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *WWW*. 173–182.

[22] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. 2014. Practical Lessons from Predicting Clicks on Ads at Facebook. In *ADKDD*. 5:1–5:9.

[23] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast Matrix Factorization for Online Recommendation with Implicit Feedback. In *SIGIR*. 549–558.

[24] Jonathan L Herlocker, Joseph A Konstan, and John Riedl. 2000. Explaining collaborative filtering recommendations. In *CSCW*. 241–250.

[25] K. Hornik, M. Stinchcombe, and H. White. 1989. Multilayer Feedforward Networks Are Universal Approximators. *Neural Networks* 2, 5 (1989), 359–366.

[26] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. 77–118.

[27] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. 2017. Model Ensemble for Click Prediction in Bing Search Ads. In *WWW*. 689–698.

[28] Julian J. McAuley and Jure Leskovec. 2013. Hidden factors and hidden topics: understanding rating dimensions with review text. In *RecSys*. 165–172.

[29] Liqiang Nie, Meng Wang, Zheng-Jun Zha, and Tat-Seng Chua. 2012. Oracle in Image Search: A Content-Based Approach to Performance Prediction. *TOIS* 30, 2 (2012), 13:1–13:23.

[30] Liqiang Nie, Shuicheng Yan, Meng Wang, Richang Hong, and Tat-Seng Chua. 2012. Harvesting visual concepts for image search with complex queries. In *MM*. 59–68.

[31] Zhaochun Ren, Shangsong Liang, Piji Li, Shuaiqiang Wang, and Maarten de Rijke. 2017. Social Collaborative Viewpoint Regression with Explainable Recommendations. In *WSDM*. 485–494.

[32] Steffen Rendle. 2010. Factorization machines. In *ICDM*. 995–1000.

[33] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*. 452–461.

[34] Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2010. Factorizing Personalized Markov Chains for Next-basket Recommendation. In *WWW*. 811–820.

[35] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *WWW*. 285–295.

[36] Ying Shan, T. Ryan Hoens, Jian Jiao, Haijing Wang, Dong Yu, and JC Mao. 2016. Deep Crossing: Web-Scale Modeling Without Manually Crafted Combinatorial Features. In *KDD*. 255–262.

[37] Amit Sharma and Dan Cosley. 2013. Do social explanations work?: studying and modeling the effects of social explanations in recommender systems. In *WWW*. 1133–1144.

[38] Nava Tintarev. 2007. Explanations of recommendations. In *RecSys*. 203–206.

[39] Nava Tintarev and Judith Masthoff. 2011. Designing and evaluating explanations for recommender systems. *Recommender Systems Handbook* (2011), 479–510.

[40] Jesse Vig, Shilad Sen, and John Riedl. 2009. Tagsplanations: explaining recommendations using tags. In *IUI*. 47–56.

[41] Meng Wang, Weijie Fu, Shijie Hao, Hengchang Liu, and Xindong Wu. 2017. Learning on Big Graph: Label Inference and Regularization with Anchor Hierarchy. *TKDE* 29, 5 (2017), 1101–1114.

[42] Meng Wang, Weijie Fu, Shijie Hao, Dacheng Tao, and Xindong Wu. 2016. Scalable Semi-Supervised Learning by Efficient Anchor Graph Regularization. *TKDE* 28, 7 (2016), 1864–1877.

[43] Suhang Wang, Charu Aggarwal, and Huan Liu. 2017. Randomized Feature Engineering As a Fast and Accurate Alternative to Kernel Methods. In *SIGKDD*. 485–494.

[44] Xiang Wang, Xiangnan He, Liqiang Nie, and Tat-Seng Chua. 2017. Item Silk Road: Recommending Items from Information Domains to Social Users. In *SIGIR*. 185–194.

[45] Xiang Wang, Liqiang Nie, Xuemeng Song, Dongxiang Zhang, and Tat-Seng Chua. 2017. Unifying Virtual and Physical Worlds: Learning Toward Local and Global Consistency. *TOIS*. 36, 1 (2017), 4:1–4:26.

[46] Jun Xiao, Hao Ye, Xiangnan He, Hanwang Zhang, Fei Wu, and Tat-Seng Chua. 2017. Attentional Factorization Machines: Learning the Weight of Feature Interactions via Attention Networks. In *IJCAI*. 3119–3125.

[47] Xiao Yu, Xiang Ren, Yizhou Sun, Quanquan Gu, Bradley Sturt, Urvashi Khandelwal, Brandon Norick, and Jiawei Han. 2014. Personalized entity recommendation: a heterogeneous information network approach. In *WSDM*. 283–292.

[48] Yongfeng Zhang, Guokun Lai, Min Zhang, Yi Zhang, Yiqun Liu, and Shaoping Ma. 2014. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *SIGIR*. 83–92.

[49] Qian Zhao, Yue Shi, and Liangjie Hong. 2017. GB-CENT: Gradient Boosted Categorical Embedding and Numerical Trees. In *WWW*. 1311–1319.